PHILIPS

MICROPROCESSOR ZELFSTUDIE-CURSUS

HET INSTRUCTIE-PAKKET VAN DE MICROPROCESSOR 2650

3

HET INSTRUCTIE-PAKKET VAN DE MICROPROCESSOR 2650

INHOUD

Het programmeren van microprocessoren heeft t.o.v. grote processoren voor algemene doeleinden enkele bijzondere kenmerken. Als eerste bijzondere kenmerk kan worden vermeld dat een programma, dat voor de microprocessor wordt geschreven en in het geheugen geregistreerd staat, meestal niet uitwisbaar mag zijn, omdat de microprocessor b.v. in een apparaat is ingebouwd. Met het uitvallen van de stroomvoorziening zou dan bij een Random Access Memory (RAM) het programma verdwijnen; het apparaat kan dan niet meer opnieuw in gebruik worden genomen. Het programma is daarom vaak in een z.g. Read Only Memory (ROM) aanwezig, terwijl voor het registreren van tussenresultaten voor snelle communicatie met de buitenwereld, veelal een RAM beschikbaar is. Voorts hebben veel microprocessoren een woordlengte van 8 bits, een zogenaamde byte. Nu is de verzameling instructies van een microprocessor, gezien zijn beperkingen, groot ten aanzien van de instructieverzameling van grotere rekenmachines. Bij kleine woordlengte houdt dit in dat men veelal dient te woekeren met het aantal bits dat per operatiecode aanwezig kan zijn. Voorts vergt de adressering van data bij elke machine over het algemeen vrij veel bits in het adresdeel van een instructie. Naast de operatiecode zijn veelal een aantal speciale bits aanwezig voor indirecte adressering, indexering e.d. Dit heeft tot gevolg dat voor het adresseren van instructies en data het aantal bits in de rest van de instructie beperkt blijven. Men moet het instructieformaat van een microprocessor dan ook vaak in het kader van dit aspect beschouwen. In het hierna volgende gedeelte zullen we op deze beperkingen nader ingaan.

De instructieverzameling van de microprocessor 2650 heeft een aantal belangrijke instructies op rekenkundig en logisch gebied en voor de behandeling van de in- en uitvoer van data. De instructieverzameling telt 75 verschillende instructies.

Teneinde een zo groot mogelijk geheugen te kunnen gebruiken, en de datastroom in de processor, het geheugen en de in- en uitvoerapparatuur zo efficiënt mogelijk te begeleiden, zijn acht adresseermethoden mogelijk. De processor heeft de mogelijkheid om te indexeren. Voorts kan het verhogen of het verlagen van de index met 1, in één instructie gerealiseerd worden. De instructies zijn 1, 2 of 3 bytes groot, de instructies voor de in- en uitvoer van data slechts 1 of 2 bytes. Het is mogelijk om in de processor afzonderlijke bits van registers te onderzoeken. Dit alles heeft tot gevolg dat een programma in een zo klein mogelijk geheugen kan worden geregistreerd, hetgeen bij de toepassing van de microprocessor tot besparing leidt.

INSTRUCTIEFORMATEN BIJ DE MICROPROCESSOR 2650

Bij moderne processoren zijn veel instructies op een bepaalde wijze ingedeeld, d.w.z. dat bepaalde bits verantwoordelijk zijn voor operatiecodes, voor bepaalde registerkeuzes, indirecte adressering, enz. Een belangrijke rol speelt uiteraard de adressering. De microprocessor 2650 kan op acht wijzen adresseren. Deze zijn:

register-adressering
immediate adressering
relatieve adressering
relatieve/indirecte adressering
absolute adressering
absolute/indirecte adressering
absolute/geïndexeerde adressering
absolute/indirecte/geïndexeerde adressering

We zullen nu enkele belangrijke formaten van instructies bespreken.

Register-adressering

De instructies die men onder register-adressering kan rangschikken, hebben betrekking op twee operanden. De eerste operand is het register 0. De tweede operand is in een van de registers. Dit register kan ook het register 0 zelf zijn. Bit 0 en bit 1 van dit type instructie zijn bestemd om het registernummer in te vullen, zie fig. 1. Het invullen van een 00 betekent dat register 0 dienst doet voor beide operanden en dat de informatie vervangen wordt door het resultaat van de operatie. Wanneer 01 is aangegeven, dan vindt de operatie plaats tussen de inhoud van register 0 en register 1, bij 10 met de inhoud van register 0 en register 2 en bij 11 met register 0 en register 3.

Het bovenstaande geeft de indruk dat behalve register 0 nog maar drie registers aanwezig zijn. Dit is echter niet het geval. De microcomputer 2650 beschikt behalve over register 0 nog over twee maal drie registers, n.l. register 1, register 2 en register 3, en bovendien over de verzameling register 1′, register 2′ en register 3′. Er zijn dus twee registers die als register 1 worden aangeduid, twee aangeduid als register 2, en twee aangeduid als register 3. Hoe wordt onderscheid tussen deze registers gemaakt? De registers zoals hier vermeld, zijn onderverdeeld in zogenaamde banken. Bank 0 bevat de registers R1, R2 en R3, terwijl bank 1 de registers R1′, R2′ en R3′ bevat. De instructie vermeldt niet (het accent wordt niet vermeld) of er een register wordt gebruikt uit bank 0 dan wel uit bank 1. Om dit te selecteren, is een aparte instructie nodig, namelijk de instructie BANK SELECT, waarover later meer. We kunnen nu ermee volstaan te vermelden dat deze instructie de bit RS (Register Select) van het Program Status Word (PSW) beïnvloedt en, afhankelijk van de inhoud van deze bit, wordt hetzij bank 0 dan wel bank 1 gekozen. In feite is register 0 een gemeenschappelijk register voor beide banken en dit kan dan ook dienst doen als communicatiemiddel om vanuit een register in bank 0 informatie over te dragen naar een register in bank 1.

Immediate adressering

Alle immediate instructies bestaan uit twee bytes. In de eerste byte is in de bits 2 t/m 7 de operatiecode aangegeven en in de bits 0 en 1 het nummer van het desbetreffende register. In dit register bevindt zich de eerste operand, en het is ook meestal de plaats waar het resultaat van de operatie geplaatst wordt. De tweede operand is aanwezig als inhoud van de tweede byte van de instructie (zie fig. 2).

Relatieve adressering

Relatieve adressering heeft betrekking op informatie in een register als de ene operand, en verdere data die in het geheugen staat geregistreerd als de andere operand. Dit heeft geen betrekking op spronginstructies, die apart behandeld zullen worden. De operatiecode is ook hier 6 bits, zie fig. 3. De bits 0 en 1 van byte 0 geven weer het register aan bij het gebruik van de registers 1, 2 en 3; er moet op worden gelet of ze zich bevinden in bank 0 dan wel in bank 1. Van byte 1 geeft bit 7 aan of het relatieve adres gebruikt dient te worden voor directe dan wel voor indirecte adressering. De bits 0 t/m 6 van byte 1 geven de relatieve verplaatsing aan t.o.v. de inhoud van het instructie-adres-register (IAR). Deze verplaatsing is gegeven in de 2-complement code. Het gebied dat bestreken wordt, strekt zich dus uit van IAR+63 naar IAR− 64.

Twee instructies hebben een andere wijze van relatieve adressering. Dit zijn de instructies:

ZERO BRANCH TO SUBROUTINE RELATIVE (ZBSR)

en

ZERO BRANCH RELATIVE (ZBRR).

Hierop wordt nog nader teruggekomen bij de behandeling van de instructies.

Absolute adressering voor niet-sprong-instructies

Alle niet-sprong-instructies zijn ter grootte van 3 bytes. Ze hebben betrekking op twee operanden waarvan één van de operandadressen geplaatst is in de bytes 1 en 2 (zie fig. 4). De bits 0 t/m 7 van byte 2 geven het minst significante deel van het desbetreffende adres van de tweede operand, de bits 0 t/m 4 van byte 1 het meest significante deel. Er zijn dus 13 bits beschikbaar voor de adressering, d.w.z. het is mogelijk 8K te adresseren; dat is een pagina. Het paginanummer wordt ontleend aan de twee meest significante bits van het IAR.

Bit 7 van byte 1 geeft aan of de tweede operand direct dan wel indirect geadresseerd moet worden. Bij directe adressering is het zojuist gegeven adres het absolute adres. Indien bit 7 van byte 1 een 1 is, dan wordt gebruik gemaakt van de indirecte adressering en verwijst het adres in de instructie naar de eerste van twee geheugenplaatsen waar men het adres van de gewenste operand kan aantreffen. De tweede operand is dan nog niet ondubbelzinnig bepaald. De bepaling hangt af van de inhoud van de bits 5 en 6 van byte 1. Men noemt deze bits de z.g. Index Control (IC). Zijn bits 5 en 6 beide 0, dan heeft er geen indexering plaats. De eerste operand wordt dan gegeven door de bits 0 en 1 van byte 0 en deze hebben dan betrekking op een register. Hierbij moet uiteraard weer in acht worden genomen in welke registerbank men werkzaam is.

Indien van byte 1 de bits 5 en 6 niet beide 0 zijn, dan vindt er indexering plaats.

De eerste operand bevindt zich nu bij dit soort instructies in register 0.

De bits 0 en 1 van byte 0 geven dan aan welk register als INDEX REGISTER wordt gebruikt. In het geval dat bit 5 en bit 6 van byte 1 beide gelijk zijn aan 1, heeft er een

eenvoudige indexering plaats. Als bit 6 =0 en bit 5 =1, dan heeft er eveneens indexering plaats, maar wordt vóór de uitvoering van de instructie de index eerst met 1 verhoogd. Als bit 6 =1 en bit 5 =0, dan wordt het Index Register met 1 verlaagd. Zie onderstaande tabel.

Index Control (IC)

Bit 6	Bit 5	Betekenis
0	0	Niet geïndexeerd adres
0	1	Indexering en index 1 verhogen
1	0	Indexering en index 1 verlagen
1	1	Indexering

Als de indexen verhoogd (resp. verlaagd) worden, dan gebeurt dit alvorens de inhoud van het Index Register gebruikt wordt voor de berekening van het nieuwe adres.

Bij indirecte adressering verwijst het desbetreffende adres naar de plaats waar het adres van de operand aanwezig is. De indexering, indien aanwezig, geschiedt nu ten opzichte van dit laatste adres.

Bij elke adresbepaling m.b.v. indexering wordt de inhoud van het Adres Register beschouwd als een 8 bits lang absoluut geheel getal. Er wordt dus niet in het 2-complement talstelsel gewerkt. Men kan m.b.v. het Index Register dus uitsluitend naar adressen kijken die hoger zijn dan de basis; deze wordt aangegeven door het adres in byte 2 en 3 of in het veld met het indirecte adres (maximaal 255 bytes).

Toelichting: Fig. 5 geeft een voorbeeld van een absolute adressering, waarbij niet geïndexeerd wordt en ook geen gebruik gemaakt wordt van een indirect adres. De eerste operand is in register XX, de tweede operand is op de plaats Y..Y. De adressering van deze tweede operand geschiedt in dezelfde pagina als de instructie zich bevindt. In fig. 5 is ook aangegeven hoe het resultaat wordt berekend. In dit geval is de eerste operand R[XX] en de tweede operand MEM[PAG,Y..Y]. De waarde XX tussen haken geeft de index weer binnen de groep van de registers, d.w.z. het nummer van het desbetreffende register waarin zich de eerste operand bevindt. De haken achter MEM geven aan dat eerst het paginanummer dient te worden genomen en dat vervolgens de waarde Y..Y daaraan moet worden toegevoegd. Dat het resultaat van de operatie in het register [XX] moet worden geplaatst, wordt aangegeven door een pijl ←.

In fig. 6 is een voorbeeld van een absolute adressering bij niet-sprong-instructies waarbij geen indexering plaats heeft, maar wèl de indirecte adressering. Dit wordt kenbaar gemaakt doordat in byte 1 de bits 7 t/m 5 resp. zijn 1, 0 en 0. Het adres Y (bits 0 t/m 4 in byte 1 en bits 0 t/m 7 in byte 2) verwijst nu naar een adres in dezelfde pagina. Op dit adres vindt men het meest significante deel van het effectieve adres en in de daarop volgende byte het minst significante deel.

Van de 16 bits van dit effectieve adres worden er 15 gebruikt voor de adressering van het gewenste tweede operand. Onderaan de bladzijde vindt men wederom de uitdrukking voor de berekening. Hier ziet men een dubbele indexering binnen het geheugen. De hoofdindex, gegeven door de buitenste haken, wordt weergegeven door een geheugenplaats met zijn eigen index, namelijk MEM[MEM[PAG, Y..Y]. Dit is de wijze van aangeven van de indirecte adressering. Het effectieve adres wordt gevonden op de plaats MEM[PAG,Y..Y].

In fig. 7 is de absolute adressering aangegeven, waarbij indexering plaats heeft en geen indirecte adressering, d.w.z. dat van byte 1 de bits 5, 6 en 7 resp. zijn 1, 1 en 0. De bits, ingevuld met de Y, geven het adres aan binnen de pagina waarin de instructie werkzaam is, d.w.z. de verplaatsing t.o.v. de ondergrens van deze pagina. De bits 0 en 1 van byte 0 geven nu het Index Register weer. Wat nu ontbreekt is de eerste operand.

Bij indexering is de eerste operand altijd aanwezig in register 0.

Voor het bepalen van het adres van de tweede operand wordt de inhoud van het Index Register als absoluut getal beschouwd van 8 bits groot, het kan dus elke gehele waarde bezitten in het gebied 0 t/m 255. Deze waarde wordt bij het adres Y..Y geteld en het paginanummer wordt ervoor geplaatst. Op deze wijze wordt het effectieve adres verkregen. Het resultaat van de operatie wordt in register 0 geplaatst. Een voorbeeld maakt dit duidelijk.

Stel (zie fig. 7) dat de instructie in pagina 1 aanwezig is, d.w.z. in het gebied boven de 8192. Vervolgens nemen we aan dat de inhoud van het adresdeel van de tweede operand gelijk is aan 75. Daar wordt de 01 voor geplaatst. Bij dit geheel wordt de inhoud van het indexregister R[XX] geteld. Als deze inhoud 170 was, dan zou het resultaat het effectieve adres 8437 zijn.

Als men op dezelfde wijze wenst te indexeren, maar nu m.b.v. indirecte adressering (zie fig. 8), dan past men een andere methodiek van de adresberekening toe. De inhoud van het indexregister wordt nu niet toegevoegd aan het operand-deel van de instructie. Eerst neemt men het operanddeel van de instructie, dan wordt het paginanummer ervoor geplaatst, en aldus verkrijgt men het adres waar men het beginadres kan vinden van een tweede operand. Dit beginadres wordt nu verhoogd met de inhoud van het Index Register. Dit vormt het effectieve adres waar de tweede operand zich bevindt.

Dit wordt verduidelijkt door het voorbeeld van fig. 8. Stel de instructie bevindt zich in pagina 1, d.w.z. in het stuk tussen 8K en 16K. Neem vervolgens aan dat de inhoud van het operand-deel van de instructie 75 bedraagt. Nu wordt het indirecte adres gevonden door 8192, de ondergrens van de pagina en de waarde 75 bij elkaar op te tellen. Op deze plaats bevindt zich nu het adres waar het basis-adres van de te bereiken tweede operand aanwezig is. Stel dat de inhoud

van deze plaats en de daarop volgende byte 24583 is. Bij deze waarde moet nu de inhoud van het Index Register opgeteld worden. Stel dat deze waarde, evenals in het vorige voorbeeld, 170 is. Dan is het uiteindelijke adres voor de tweede operand 24753.

Deze methode van adresseren stelt ons in staat om een bepaalde waarde uit een tabel of een bepaald element van een vector op te zoeken. Het begin van de vector of van de tabel is dan gegeven door de inhoud van de indirecte adresplaats. In ons voorbeeld dus 24583.

Het geheel is nogmaals geillustreerd in fig. 9, waar de geheugenindeling is gegeven, inclusief de inhoud van het Instructie Register, het Instructie Adres Register en het register R[XX].

Fig. 10 stelt een instructie voor waarin géén indirecte adressering is gebruikt, maar een indexering met een incrementatie van het Index Register. Zoals weergegeven, wordt eerst de inhoud van het Index Register R[XX] met 1 verhoogd. Vervolgens bepaalt men het effectieve adres door de som van het Index Register en de waarde van het adres van de tweede operand te nemen en het paginanummer ervoor te plaatsen. Vervolgens wordt de bewerking uitgevoerd tussen de eerste operand in R0 en de dat in het geheugen op de plaats die is aangegeven door het bepaalde adres. Het resultaat wordt in R0 geplaatst. De incrementatie van het register is hier kenmerkend.

Op deze wijze kan men b.v. de som van de waarden in een tabel bepalen. Elke volgende waarde wordt automatisch gevonden, daar het Index Register bij elke operatie met 1 wordt verhoogd. Na afloop van elke operatie kan men de inhoud van het Index Register vergelijken met een gegeven waarde, namelijk de lengte van de tabel. Zodra deze gelijk zijn, kan men het sommeren van de tabelwaarden beëindigen. Dit betekent b.v. het onderbreken van de lus waarin een dergelijke optelling plaats heeft. Bij de vraagstukken die bij deze handleidingen behoren, zijn hiervan voorbeelden gegeven.

Als bit 6 =1 en bit 5 =0, dan zal het indexregister vóór deze serie handelingen niet met 1 verhoogd worden, maar met 1 worden verlaagd. Nu kan men met de maximale afmeting van de tabel in het Index Register beginnen en na uitvoering van de laatste operatie zal de inhoud van het Index Register =0 zijn. Test men R[XX] hierop, dan kan men de lus waarin de operatie is opgenomen, beëindigen en het programma verder afwikkelen. Ook hiervan is in de handleiding met de vraagstukken een voorbeeld gegeven.

Bij de indexering met het auto-increment, resp. met auto-decrement, kan men uiteraard weer indirecte adressering toepassen. De gevolgde techniek is een combinatie van die van fig. 9 en fig. 10. Door middel van het indirecte adres verwijst men naar het beginpunt van de tabel; de verplaatsing in de tabel wordt aangegeven door het indexregister. Doorloopt men in een lus deze instructie, dan wordt het indexregister telkenmale verhoogd, resp. verlaagd, waardoor men element na element in de tabel als tweede operand kan gebruiken, en wel zo lang tot alle elementen aan de beurt zijn geweest.

Absolute adressering voor sprongopdrachten

Sprongopdrachten zijn noodzakelijk voor het desgewenst onderbreken van een programma, teneinde – op grond van bepaalde condities die in een processor aanwezig kunnen zijn – naar een ander programmadeel over te schakelen. Deze condities kunnen veroorzaakt worden door de uitkomst van berekeningen, maar ook doordat – bij het onderzoeken van de buitenwereld, gevolgd door de invoer van bepaalde gegevens – een sprong gemaakt wordt. Ook komt het voor dat de buitenwereld de microprocessor om een speciale dienstverlening verzoekt, en dat de processor voor deze dienstverlening gebruik maakt van een speciaal programma. Men noemt zulk een verzoek een interruptie. De sprong die de machine op een bepaald moment uitvoert, hangt af van de conditie van de processor. Deze conditie kan zijn weerslag vinden in het Conditie Code register CC, dat een onderdeel vormt van het Program Status Word (PSW), en wel de bits PSL 6 en PSL 7. De samenhang van deze conditiecode CC met het masker, gegeven in de instructie, wordt bij de afzonderlijke spronginstructies behandeld.

In byte 0 van de spronginstructie vindt men in de bits 2 t/m 7 de operatiecode en in de bits 0 en 1 ervan de conditievoorwaarden. Bit 7 van byte 1 doet weer dienst om aan te geven of het adres waarnaar men wenst te springen, direct gegeven is (bit 7 =0), dan wel indirect te bereiken is (bit 7 =1). De overige 7 bits van byte 1 en de 8 bits van byte 2 vormen samen een adres van 15 bits, hetgeen voldoende is om elke plaats in een 32K geheugen te adresseren. Door de indirecte adressering is een beperkte dynamische adressering mogelijk. De instructie die in een ROM aanwezig is, kan verwijzen naar een adres in een RAM. Daar de inhoud van dit adres gewijzigd kan worden, is het heel goed mogelijk, afhankelijk van het verloop van het proces, de inhoud van deze geheugencellen te wijzigen en daarmee, bij de uitvoering van de spronginstructie, een sprong te maken naar een bepaalde programmamodule.

Instructies van diverse typen

In de instructieverzameling van de microprocessor 2650 komen nog enkele andere instructies voor, die verschillend van aard zijn. Een voorbeeld hiervan is de instructie die zorg draagt voor het beëindigen van het programma, (de z.g. HALT-instructie). Ook is er een instructie waarbij wel een zekere tijd verloopt, maar waarbij géén bepaalde operatie wordt uitgevoerd (de z.g. NOP-instructie). Deze instructies laten zich slecht classificeren en ze hebben dan ook géén bepaalde benaming gekregen.

DE INSTRUCTIEVERZAMELING

In dit deel worden de instructies die voorkomen in de instructieverzameling van de microprocessor 2650, elk afzonderlijk behandeld. Bij de behandeling zullen we aangeven wat het resultaat zal zijn ten aanzien van de Conditie Code na afloop van de operatie, en tevens hoe veel klokperioden een operatie in beslag neemt. Dit onder voorbehoud dat het bijbehorende geheugen snel genoeg is om de vereiste informatie te verstrekken, resp. in ontvangst te nemen. Zie hiervoor de handleiding voor de "HARDWARE VAN DE 2650 MICROPROCESSOR".

De instructies worden niet in een bepaalde volgorde behandeld wat betreft de grootte of de soort. De volgorde van behandeling is uitsluitend gericht op het geleidelijk opbouwen van de kennis omtrent de instructies, mede aan de hand van vraagstukken die in de handleiding "VRAAGSTUKKEN" zijn gegeven. De lezer wordt dringend aangeraden om deze vraagstukken niet over te slaan, daar ze vaak goede vingerwijzingen bevatten voor de toepassing van de instructies.

Heeft men de opgaven die bij de diverse instructies behoren, volbracht, dan is het bijzonder nuttig om ze onmiddellijk te proberen op de INSTRUCTOR 50.

De lezer doet er dan ook goed aan, deze handleiding nu te onderbreken en verder te gaan met de handleiding over de "INSTRUCTOR 50. Heeft hij deze gelezen tot blz. 10, dan dient hij weer naar deze handleiding terug te keren en de instructies verder te bestuderen. Na een aantal instructies wordt opnieuw naar de handleiding over de INSTRUCTOR 50 verwezen met de raad aan de lezer het desbetreffende deel van deze handleiding dan weer te bestuderen.

Men zou deze werkwijze, in het kader van het programmeren, kunnen beschouwen als het springen naar een subroutine.

Algemene opmerkingen:

Bij elke instructie zijn de z.g. MNEMONICS vermeld. Dit zijn symbolische afkortingen van de instructienamen, die het onthouden van deze instructies vergemakkelijken. Als twee operanden in de instructie voorkomen, dan wordt de eerste operand, meestal een register, onmiddellijk na deze code vermeld, en daarvan gescheiden door een komma. Na enige spaties wordt de indicatie voor de tweede operand vermeld. Deze tweede operand kan een getal zijn of een masker (immediate adressering), dan wel een adres. Zie onderstaand voorbeeld:

```
LODI,r    data
LODR,r    adres
```

Desgewenst kan men dit adres invullen, resp. vervangen door een LABEL. Deze laatste methode wordt sterk aanbevolen omdat men zich hier nog niet bindt aan bepaalde plaatsen van instructies en data.

Indirecte adressering geeft men aan door het plaatsen van een asterisk (*) vóór dit adres. Indien een instructie geschikt is om indirect te worden geadresseerd, dan wordt na de mnemonic code en vóór dit adres een asterisk geplaatst.

Een index wordt aangegeven m.b.v. het nummer van het desbetreffende register achter het adres van de operand. De beide worden door een komma gescheiden. Bij de behandeling van de instructies treft men (,X) aan achter het adres als indicatief van een indexeermogelijkheid. De haken betekenen hier dat X facultatief aangegeven kan worden. Het weglaten van de komma en een registernummer impliceert dan dat er niet geïndexeerd wordt. Slaat men op de INSTRUCTOR 50 de code aan, dan behoren bit 5 en bit 6 van de tweede byte in dat geval een 0 te zijn.

Voorbeeld:

LODA,r (*)adres(,X) (,±)

Dit betekent, dat zowel de *, de X en de ± facultatief zijn. Bij het weglaten (default) moeten in de oorspronkelijke instructie nullen ingevuld worden op de daarvoor bestemde plaatsen. De ± geven een auto-increment/-decrement aan.

Enkele voorbeelden met de bovenstaande instructie zijn:

	Mnemonic		Byte 0	Bytes 1 en 2
1.	LODA,1	adres	H'0D'	000<adres>
2.	LODA,1	*adres	H'0D'	100<adres>
3.	LODA,0	adres,R2	H'0E'	000<adres>
4.	LODA,0	*adres,R2,+	H'0E'	101<adres>

ad 1. Een Laad-Absoluut-instructie zonder indexering en directe adressering.

ad 2. Een Laad-Absoluut-instructie zonder indexering en indirecte adressering.

ad 3. Een Laad-Absoluut-instructie met indexering (R2) en directe adressering (eerste operand in R0).

ad 4. Een Laad-Absoluut-instructie met indexering (R2 en eerste operand in R0) en indirecte adressering en auto-increment.

Een verklaring van de gebruikte symboliek wordt hieronder gegeven bij de instructies.

Verklaring van de gebruikte symboliek bij de instructies

R[0] ← R[1]	De inhoud van register 1 wordt geplaatst in register 0. De inhoud van register 1 blijft ongewijzigd.

IAR ← IAR + 1

De inhoud van het IAR wordt met 1 vermeerderd en geplaatst in het IAR. Effectief betekent dit dat het adres in het IAR met 1 wordt verhoogd.

R[x] ← v

v wordt in het register x geplaatst.

R[x] ← MEM[ADRES]

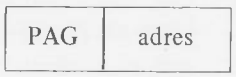
In het register x wordt de inhoud van het woord in het geheugen dat zich op de plaats ADRES bevindt, geplaatst.

ADR ← MEM[IAR + ADRES]

Eerst wordt de som bepaald van de inhoud van het IAR en het gegeven adres. Deze som wordt als adres gebruikt om data uit het geheugen te verkrijgen. Deze data wordt toegekend aan ADR, die een (tijdelijke) variabele is en dienst zal doen als effectief adres.

ADR ← (PAG,adres) + R[r]

Het paginanummer wordt geplaatst vóór het "adres". Ze vormen samen aldus een adres dat voor het geheugen bruikbaar is:



2 bits 13 bits

Vervolgens wordt deze waarde verhoogd met de inhoud van register r en aan ADR toegekend.

ADR ← MEM[PAG,adres] + R[r]

Eerst wordt een adres gevormd door het paginanummer en het "adres" aaneen te rijgen, zoals in het vorige voorbeeld. Deze waarde doet dienst als adres om in het geheugen twee bytes op het gegeven adres en adres+1 te lezen. Vervolgens wordt deze waarde met de inhoud van register r verhoogd. De nieuwe waarde wordt aan ADR toegekend.

Wijze van coderen

Wenst men een programma te schrijven, dan kan men het best in het geheel geen absolute adressen gebruiken, maar uitsluitend symbolische namen, zogenaamde LABELS. Op de bijgevoegde bladen treft men deze indeling aan. Heeft men b.v. een LAAD-instructie, dan moet men zo veel regels open laten als de instructie meer bytes telt dan 1 byte, m.a.w. bij een instructielengte van 3 bytes laat men twee regels open. Als men het programma op deze wijze schrijft, dan kan men later in de linkerkolom de desbetreffende waarden invullen.

Als voorbeeld geven we hier het label LADEN, dat op het adres 0005 staat. Het label PUNTW staat op plaats 0017, zodat men bij het coderen van de mnemonic code in de absolute code, deze waarden op eenvoudige wijze kan invullen. Het voorbeeld spreekt wat dat betreft voor zichzelf. De 17 kan zonder meer als byte 2 worden overgenomen. De byte 1 is 00 wegens het meest significante deel van het adres, maar wegens de indirecte adressering moet er 8 aan de meest hexadecimale waarde worden toegevoegd, zodat byte 1 80 wordt. De index bevindt zich in register 2, zodat hier de operatiecode 0E moet worden gebruikt.

Men doet er verstandig aan, bij de symbolische adressering, zoals in dit geval gegeven, achter een LODA-instructie een ,0 te plaatsen, waarmee men duidelijk aangeeft dat register 0 de eerste operand is, waarop het laden betrekking heeft, namelijk het doel waar de informatie naar toegebracht wordt. Dit maakt het lezen van een programma eenvoudiger. Het moet wel duidelijk zijn dat de 2 van de tweede operand het werkelijke register is dat in de operatiecode wordt vermeld, vandaar dat byte 0 0E bedraagt. Byte 1 zal wegens de indirecte adressering ten minste 80 zijn. Voorts wordt hieraan toegevoegd de waarde + wegens het increment en verder geen waarde, omdat de twee significante hexadecimale digits van het adres beide 0 zijn. Hierdoor wordt voor byte 1 de waarde A0 gekregen. Byte 2 is identiek aan de minst significante hexadecimale digits van het adres, dus 17.

De aankomende programmeur wordt dringend aangeraden deze methodiek te gebruiken omdat dit de veiligste manier is om te vermijden dat hij onherroepelijk in fouten vervalt. Het correleren tussen de fysische adressen en de instructies in mnemonic code is normaal gesproken de taak van een assembler, die daarvoor een aparte teller heeft. Deze telling kan men natuurlijk achteraf zelf verrichten door ze bij te houden. Ze wordt echter automatisch gegeven als men de blanko ruimte tussen de instructies voegt.

BESCHRIJVING	FASE 1	BLADZ.

REGEL	ADRES	DATA B0	DATA B1	DATA B2	LABEL	SYMBOLISCHE INSTRUCTIE OPCODE	SYMBOLISCHE INSTRUCTIE OPERANDEN	COMMENTAAR
1	0005	0E			LADEN	LODA,0	*PUNTW,2,+	laadinstructie, index-register 2 met incre-
2	0006					- -		ment en indirecte adressering
3	0007					- -		
4	0008				VOLGENDE INSTRUCTIES			
5	0009					- -		
6	000A					- -		
7	000B					- -		
8	000C					- -		
9	000D					- -		
10	000E					- -		
11	000F					- -		
12	0010					- -		
13	0011					- -		
14	0012					- -		
15	0013					- -		
16	0014					- -		
17	0015					- -		
18	0016					- -		
19	0017				PUNTW	adres MS byte		
20	0018					adres LS byte		
21								
22								
23								
24								
25								
26								

DIRECTE RELATIEVE ADRESSERING: TWEEDE BYTE	+	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F –
	N	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 –
	–	7F 7E 7D 7C 7B 7A 79 78 77 76 75 74 73 72 71	70 6F 6E 6D 6C 6B 6A 69 68 67 66 65 64 63 62 61 –
TEL H'80' BIJ DE AFWIJKING OM HET INDIRECTE ADRES TE BEPALEN	+	20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F	30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F –
	N	32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47	48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64
	–	60 5F 5E 5D 5C 5B 5A 59 58 57 56 55 54 53 52 51	50 4F 4E 4D 4C 4B 4A 49 48 47 46 45 44 43 42 41 40

BESCHRIJVING	FASE 2	BLADZ.

REGEL	ADRES	DATA B0	B1	B2	LABEL	SYMBOLISCHE INSTRUCTIE OPCODE	OPERANDEN	COMMENTAAR
1	0005	0E			LADEN	LODA,0	*PUNTW,2,+	laadinstructie, index-register 2 met incre-
2	0006		A0			--		ment en indirecte adressering
3	0007			17		--		
4	0008				VOLGENDE INSTRUCTIES			
5	0009					--		
6	000A					--		
7	000B					--		
8	000C					--		
9	000D					--		
10	000E					--		
11	000F					--		
12	0010					--		
13	0011					--		
14	0012					--		
15	0013					--		
16	0014					--		
17	0015					--		
18	0016					--		
19	0017				PUNTW	adres MS byte		
20	0018					adres LS byte		
21								
22								
23								
24								
25								
26								

DIRECTE RELATIEVE ADRESSERING: TWEEDE BYTE	+ 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F N 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 – 7F 7E 7D 7C 7B 7A 79 78 77 76 75 74 73 72 71	10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F – 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 – 70 6F 6E 6D 6C 6B 6A 69 68 67 66 65 64 63 62 61 –
TEL H'80' BIJ DE AFWIJKING OM HET INDIRECTE ADRES TE BEPALEN	+ 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F N 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 – 60 5F 5E 5D 5C 5B 5A 59 58 57 56 55 54 53 52 51	30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F – 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 50 4F 4E 4D 4C 4B 4A 49 48 47 46 45 44 43 42 41 40

Binaire Code

1	1	0	0	0	0	0	0
7	6	5	4	3	2	1	0

Executietijd: 2 machinecycli (6 klokperioden)

PSW-bits die beïnvloed worden: geen

Beschrijving

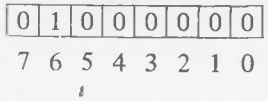
Deze instructie van 1 byte draagt ervoor zorg dat er in de processor geen enkele actie wordt ondernomen. Er worden geen registers veranderd, noch wordt de conditiecode beïnvloed. Het enige wat plaats heeft, is het niet actief zijn gedurende 6 klokperioden.

Opmerking:

Behalve als functionele code, vooral in tijdlussen, is deze NOP-instructie bijzonder waardevol voor hen die beginnen met het programmeren in de hexadecimale code van de microprocessor 2650. Als men een programma geschreven heeft, dan dient men alle labels te vervangen door de werkelijke adressen die voor de betreffende labels gelden. Heeft men tijdens het programmeren een fout gemaakt, dan kan het zijn dat een instructie moet worden tussengevoegd, dan wel dat een instructie moet worden weggelaten. Dit zal tot gevolg hebben dat men alle adressen opnieuw moet bepalen, zodat het nieuwe programma in zijn geheel weer in de INSTRUCTOR gebracht dient te worden. Als men een instructie moet weglaten omdat deze overbodig is, kan men deze eenvoudig vervangen door één of meer (afhankelijk van de lengte van de instructie) NOP-instructies. Voorts is het zinvol bij het schrijven van een programma, een aantal NOP-instructies op diverse plaatsen achtereenvolgens op te nemen. Moet men namelijk nog een instructie toevoegen, dan kan men de NOP-instructies laten vervallen en vervangen door de nu noodzakelijke instructie, zonder dat er iets aan de adressen behoeft te worden veranderd. Speciaal voor beginnende programmeurs is de NOP-instructie, met het oog op het maken van programmeerfouten, bijzonder nuttig.

De operatiecode luidt:
H' C0'

Binaire Code



Executietijd: 1 cyclus (3 klokperioden)

PSW-bits die beïnvloed worden: geen

Beschrijving
Als de 2650 microprocessor deze instructie uitvoert, zal geen volgende instructie uit het geheugen worden betrokken. De RUN/WAIT-pen wordt in de wachttoestand gezet. Men kan deze toestand beëindigen door een RESET-signaal.

Opmerking:
met behulp van deze instructie kan men een programma beeindigen en dit via de bovengenoemde pen aan de omgeving kenbaar maken. Zie de handleiding over de "HARDWARE VAN DE 2650 MICROPROCESSOR".

Als er een INTERRUPT toegestaan is, dan zal het optreden van een INTERRUPT de processor opnieuw in werking stellen. Welke instructie dan uitgevoerd wordt, hangt af van het interruptprogramma; dit wordt bij INTERRUPT behandeld.

De operatiecode luidt:
H' 40'

Verricht thans oefening 12!

Binaire code

0	0	0	0	0	0	r
7	6	5	4	3	2	1 0

Executietijd: 2 cycli (6 klokperioden)

PSW-bits die beïnvloed worden: CC

Beïnvloeding Conditie-Code:

register 0	CC1	CC0
positief	0	1
nul	0	0
negatief	1	0

IAR ← IAR+1
R[0] ← R[r]

Beschrijving

De instructie is 1 byte groot en zorgt voor de belangrijke communicatie in de hoogste orde in de geheugen-hiërarchie, namelijk tussen het register R0 en de overige registers. Onder de overige registers worden verstaan de registers R1, R2 en R3 of R1′, R2′ en R3′, afhankelijk of ze aanwezig zijn in bank 0 resp. bank 1. Voor de goede orde zij vermeld dat de accenten ′ van R1′ enz. niet in de instructie worden aangebracht: dit moet gebeuren door vóór het uitgeven van deze instructie d.m.v. de PSL-bit 4 (RS) de juiste register-bank te kiezen (dit wordt later behandeld). Het register R0 is van belang als communicatiecentrum. Het overbrengen van informatie van de registers tussen bank 0 en bank 1 kan via R0 geschieden; dit is althans de kortste methode. Een andere methode is uiteraard die via het geheugen, maar deze wijze neemt meer tijd in beslag.

Ook doet register R0 dienst als communicatiecentrum voor het laden, resp. lezen van de beide delen van het PSW en de I/O. Dit register is dus van groot belang.

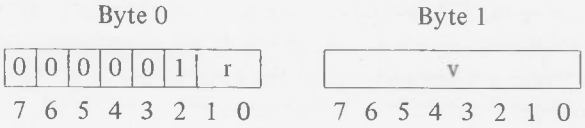
De operatiecodes luiden:

R[0] ← R[0] H′ 00′
R[0] ← R[1] H′ 01′
R[0] ← R[2] H′ 02′
R[0] ← R[3] H′ 03′

Deze codering treft men ook aan op de witte strip van de INSTRUCTOR. De registers, resp. eventuele conditiecodes zijn horizontaal aangegeven, terwijl verticaal de instructies vermeld zijn. Op het kruispunt vindt men de betreffende hexadecimale code voor de operatie.

Verricht oefening 13!

Binaire Code



Executietijd: 2 cycli (6 klokperioden)

PSL-bits die beïnvloed worden: CC

Beïnvloeding Conditie-Code:

Register r	CC1	CC0
positief	0	1
nul	0	0
negatief	1	0

IAR ← IAR+2
R[r] ← v

Beschrijving

De LOAD IMMEDIATE instructie is 2 bytes groot. De tweede byte bevat een waarde/vector. Deze "v" kan een getal zijn óf een masker dat nodig is voor later gebruik. De inhoud van het gekozen register r gaat verloren en wordt vervangen door de waarde v.

Deze instructie is vooral van waarde in die gevallen waarin men een vector/getal wenst te gebruiken, zonder dat dit in het RAM behoeft te staan. Het is een van de kortste wijzen van adresseren. Elke andere vorm van een adressering die ten doel heeft één byte uit het geheugen te verkrijgen, vraagt meer ruimte voor instructies. Voor het inbrengen van constanten en logische vectoren is de LOAD IMMEDIATE de meest aangewezen methode.

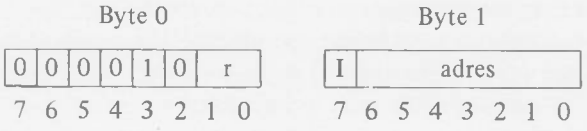
De operatiecodes luiden:

R[0] ← v	H' 04'
R[1] ← v	H' 05'
R[2] ← v	H' 06'
R[3] ← v	H' 07'

Achter deze instructies moet als tweede byte de gewenste vector worden ingevuld.

Verricht nu oefening 14!

Binaire Code



Executietijd: 3 cycli (9 klokperioden)
5 cycli (15 klokperioden) bij indirecte adressering

PSW-bits die beïnvloed worden: CC

Beïnvloeding Conditie-Code:

register r	CC1	CC0
positief	0	1
nul	0	0
negatief	1	0

Directe adressering
I=0
IAR ← IAR+2
ADR ← IAR+adres
R[r] ← MEM[ADR]

Indirecte adressering
I=1
IAR ← IAR+2
ADR ← MEM[IAR+adres]
R[r] ← MEM[ADR]

Beschrijving

Deze instructie ter grootte van 2 bytes verplaatst één byte uit het geheugen naar het aangegeven register. De CC zal de waarde aangeven van de inhoud van het register (in 2-complement). Veelal zal deze instructie gebruikt worden met de indirecte adressering. Immers als vaak dezelfde variabele moet worden geladen in verschillende instructies, dan spaart men geheugenruimte door het adres van deze variabele op een vaste plaats in het geheugen onder te brengen. Door middel van de relatieve adressering, gevolgd door indirecte adressering, kan men de betreffende variabele eenvoudig in twee stappen vinden. Hierdoor wordt per instructie 1 byte bespaard.

De operatiecodes luiden:
voor register R0 H'08'
voor register R1 H'09'
voor register R2 H'0A'
voor register R3 H'0B'

Na deze operatiecodes dient men vervolgens het adres aan te geven, en eventueel de indicatie voor de indirecte adressering. Dit laatste komt er op neer dat men 8 optelt bij het meest significante hexadecimale deel van het adres.

Verricht nu de oefeningen 15 en 16!

Binaire code

Byte 0

0	0	0	0	1	1	r/X
7	6	5	4	3	2	1 0

Byte 1

I	IC	adres High Order
7	6 5	4 3 2 1 0

Byte 2

adres Low Order
7 6 5 4 3 2 1 0

Executietijd: 4 cycli (12 klokperioden)
6 cycli (18 klokperioden) bij indirecte adressering

Bits van het PSW die beïnvloed worden: CC

Beïnvloeding Conditie-Code:

register r	CC1	CC0
positief	0	1
nul	0	0
negatief	1	0

Directe adressering:

```
I=0   IC=00
   IAR ← IAR+3
   ADR ← PAG, adres
   R[r] ← MEM[ADR]

I=0   IC=11
   IAR ← IAR+3
   ADR ← (PAG, adres)+R[r]
   R[0] ← MEM[ADR]

I=0   IC=01
   IAR ← IAR+3
   R[r] ← R[r]+1
   ADR ← (PAG, adres)+R[r]
   R[0] ← MEM[ADR]

I=0   IC=10
   IAR ← IAR+3
   R[r] ← R[r]−1
   ADR ← (PAG, adres)+R[r]
   R[0] ← MEM[ADR]
```

Indirecte adressering:

```
I=1   IC=00
   IAR ← IAR+3
   ADR ← MEM[PAG, adres]
   R[r] ← MEM[ADR]

I=1   IC=11
   IAR ← IAR+3
   ADR ← MEM[PAG, adres]+R[r]
   R[0] ← MEM[ADR]

I=1   IC=01
   IAR ← IAR+3
   R[r] ← R[r]+1
   ADR ← MEM[PAG, adres]+R[r]
   R[0] ← MEM[ADR]

I=1   IC=10
   IAR ← IAR+3
   R[r] ← R[r]−1
   ADR ← MEM[PAG, adres]+R[r]
   R[0] ← MEM[ADR]
```

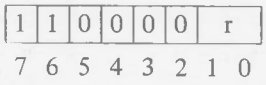
Beschrijving

Deze 3 bytes lange instructie zorgt ervoor dat een byte uit het geheugen overgedragen wordt aan een register. Het register wordt gespecificeerd in de bits 0 en 1 van byte 0 van de instructie. Dit geldt echter alleen als de bits 5 en 6 van de instructie van byte 1 beide 0 zijn, aangevende dat er niet geïndexeerd wordt. Ingeval er wèl wordt geïndexeerd (d.w.z. de bits zijn ≠0), dan is het register 0 de plaats waarnaar de desbetreffende byte wordt overgedragen; het register X doet dan dienst als indexregister. In eerste instantie, d.w.z. zonder indirecte adressering, kan elke byte binnen de pagina geadresseerd worden (het adresdeel is 13 bits lang, d.w.z. voor 1 pagina). Met indirecte adressering bevinden zich dus binnen de pagina de twee bytes van het gebied waarnaar verwezen wordt. Met behulp van het aangegeven indexregister kan men, vanaf de door het indirecte adres gegeven locatie, een bepaalde "offset" (verplaatsing) bewerkstelligen. Indirecte adressering met indexering wordt vooral gebruikt als register 0 dienst doet als communicatiecentrum voor de in- en uitvoer. Men kan nu b.v. een tabel via register 0 overhevelen naar de output.

De operatiecodes luiden:

register R0	H'0C'
register R1	H'0D'
register R2	H'0E'
register R3	H'0F'

Binaire Code



Executietijd: 2 cycli (6 klokperioden)

PSW-bits die beïnvloed worden: CC

Beïnvloeding Conditie-Code:

register r	CC1	CC0
positief	0	1
nul	0	0
negatief	1	0

IAR ← IAR+1
R[r] ← R[0]

Beschrijving

Deze instructie verplaatst de inhoud van register R0 naar het in het adresdeel aangegeven register. Als men op deze wijze een waarde verplaatst, dan zal de waarde van het CC die in de vorige instructie ontstaan is (b.v. door optelling in register R0), veranderen. Speciale aandacht moet worden geschonken aan het feit dat het register r niet identiek mag zijn aan het register 0; dit is namelijk de operatiecode 11000000, die gereserveerd is voor de operatie NOP.

Door deze STORE-operatie is een beweging van data van een register 0 naar de overige registers mogelijk, d.w.z. een datastroom in de hoogste geheugenhiërarchie met R0 als bron. Deze instructie is complementair aan de instructie LOAD REGISTER ZERO (LODZ), en de beschouwingen bij deze instructie gelden in dezelfde mate voor de hierboven vermelde instructie.

De operatiecodes luiden:

register R1	H'C1'
register R2	H'C2'
register R3	H'C3'

Binaire Code

Executietijd: 3 cycli (9 klokperioden)
5 cycli (15 klokperioden) bij indirecte adressering

PSW-bits die beïnvloed worden: geen

Directe adressering:

I=0
IAR ← IAR+2
ADR ← IAR+adres
MEM[ADR] ← R[r]

Indirecte adressering:

I=1
IAR ← IAR+2
ADR ← MEM[IAR+adres]
MEM[ADR] ← R[r]

Beschrijving

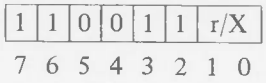
Deze instructie ter grootte van 2 bytes maakt het mogelijk, om data over te dragen naar een gebied dat in de nabijheid van de instructies gelegen is. Opgemerkt dient echter te worden dat, als een programma in een ROM staat, deze instructie alléén zinvol is bij een indirecte adressering, d.w.z. dat bit 7 van byte 1 een logische 1 dient te zijn. Voor dit geval geldt dezelfde beschouwing als voor LOAD RELATIVE (LODR). Men kan nu, als men een aantal keren vanuit een aantal instructies op één plaats de waarde van een variabele wenst in te vullen, het adres van deze variabele in het RAM plaatsen in een door het relatieve adres aangegeven locatie.

De operatiecodes luiden:

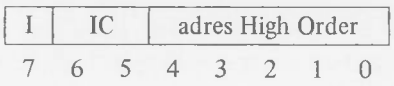
register R0	H'C8'
register R1	H'C9'
register R2	H'CA'
register R3	H'CB'

Binaire Code

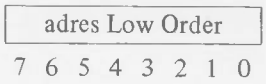
Byte 0



Byte 1



Byte 2



Executietijd: 4 cycli (12 klokperioden)
6 cycli (18 klokperioden) bij indirecte adressering

PSW-bits die beïnvloed worden: geen

Directe adressering:

I=0 IC=00
IAR ← IAR+3
ADR ← PAG, adres
MEM[ADR] ← R[r]

I=0 IC=11
IAR ← IAR+3
ADR ← (PAG, adres)+R[r]
MEM[ADR] ← R[0]

I=0 IC=01
IAR ← IAR+3
R[r] ← R[r]+1
ADR ← (PAG, adres)+R[r]
MEM[ADR] ← R[0]

I=0 IC=10
IAR ← IAR+3
R[r] ← R[r]−1
ADR ← (PAG, adres)+R[r]
MEM[ADR] ← R[0]

Indirecte adressering:

I=1 IC=00
IAR ← IAR+3
ADR ← MEM[PAG, adres]
MEM[ADR] ← R[r]

I=1 IC=11
IAR ← IAR+3
ADR ← MEM[PAG, adres]+R[r]
MEM[ADR] ← R[0]

I=1 IC=01
IAR ← IAR+3
R[r] ← R[r]+1
ADR ← MEM[PAG, adres]+R[r]
MEM[ADR] ← R[0]

I=1 IC=10
IAR ← IAR+3
R[r] ← R[r]−1
ADR ← MEM[PAG, adres]+R[r]
MEM[ADR] ← R[0]

Beschrijving

Deze instructie ter grootte van 3 bytes dient om data van een register naar het geheugen over te brengen. Indien de bits 5 en 6 van byte 1 beide 0 zijn, dan is het register aangegeven door de bits 0 en 1 van byte 0, het register waar de betreffende operand zich bevindt. Zijn de eerder genoemde bits ≠ 0, dan zal het desbetreffende register waar de operand zich bevindt, het register 0 zijn; het in de eerste byte aangegeven register is dan een indexregister. Maakt men geen gebruik van indirecte adressering dan kan elke byte binnen de pagina bereikt worden. Indien men buiten de pagina wenst te registreren, dient men van de indirecte adresseermethode gebruik te maken. Deze instructie is bijzonder waardevol als men de input via register 0 naar een gebied in het geheugen wenst over te dragen, zoals dit het geval is bij het invoeren van data via de input-poorten.

De operatiecodes luiden:

register R0	H'CC'
register R1	H'CD'
register R2	H'CE'
register R3	H'CF'

Maak de vraagstukken 17, 18, 19 en 20!

Binaire Code

1	0	0	0	0	0	r
7	6	5	4	3	2	1 0

Executietijd: 2 cycli (6 klokperioden)

PSW-bits die beïnvloed worden: C, CC, IDC, OVF

Beïnvloeding Conditie-Code:

register 0	CC1	CC0
positief	0	1
nul	0	0
negatief	1	0

IAR ← IAR+1
R[0] ← R[0]+R[r]

Indien over 8 bits heen een CARRY ontstaat, dan zal de CARRY-flip-flop in het PSW beïnvloed worden. De CARRY-bit is bit 0 van het PSL.

Beschrijving

Deze instructie ter grootte van 1 byte zorgt ervoor dat de som gevormd wordt van de inhouden van register 0 en het aangegeven register. Het resultaat wordt in register 0 geplaatst. Wordt register 0 bij register 0 opgeteld, dan komt dit neer op het vermenigvuldigen met 2. De inhoud van het register r blijft ongewijzigd.

Als 2 getallen worden opgeteld, dan zal het van de 2 meest significante bits afhangen of een CARRY ontstaat. Zijn beide bits 0, dan zal géén CARRY kunnen ontstaan en zal de CARRY-bit dus 0 worden. Is één van beide bits een 1 en het resultaat na de optelling geeft als meest significante bit een 0, dan zal er eveneens een CARRY opgetreden zijn. Zijn beide bits een 1, dan zal altijd een CARRY optreden.

Een "overflow" kan optreden als er twee negatieve of twee positieve getallen bij elkaar worden opgeteld, resp. als een positief getal en een negatief getal van elkaar worden afgetrokken.

Is het resultaat te groot (bij het optellen van 2 positieve getallen) of te klein (bij het optellen van 2 negatieve getallen), dan kan een "overflow" optreden. Deze overflow resulteert in het zetten van de bit OVF (PSL bit 2).

Bit IDC (PSL bit 5) kan eveneens gezet worden; hierop wordt nader teruggekomen bij het decimaal optellen m.b.v. de microprocessor 2650.

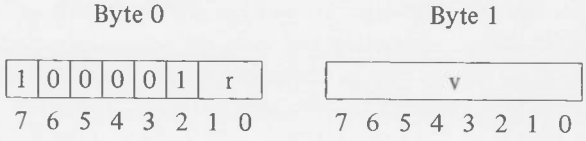
Afhankelijk van de omstandigheid of bit WC = 1 (PSL bit 3), zal bij de optelling ook de CARRY-bit C betrokken worden. In dat geval namelijk zal de inhoud van deze CARRY-bit bij de minst significante bits worden opgeteld indien de CARRY = 1 is. De CARRY-bit zelf wordt door de optelling ook beïnvloed, een en ander volgens het voorgaande. Men moet er acht op slaan dat vóór de optelling de CARRY-bit 1 kan zijn, en dus bij de optelling betrokken wordt, terwijl nà de optelling de CARRY-bit C afhankelijk is van het resultaat van de optelling. Dit is bij een multibyte getal van groot belang. Hierop wordt nog in een aparte behandeling teruggekomen.

De operatiecodes luiden:

register R0	H'80'
register R1	H'81'
register R2	H'82'
register R3	H'83'

Maak nu de vraagstukken 21 en 22!

Binaire Code



Executietijd: 2 cycli (6 klokperioden)

PSW-bits die beïnvloed worden: C, CC, IDC, OVF

Beïnvloeding Conditie-Code:

register r	CC1	CC0
positief	0	1
nul	0	0
negatief	1	0

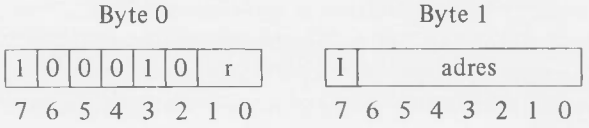
IAR ← IAR+2
R[r] ← R[r]+v

Beschrijving
Deze instructie ter grootte van 2 bytes telt, bij de inhoud van het register r, de waarde op die gegeven is door de tweede byte van de instructie. De optelling geschiedt in het 2-complement systeem. De CARRY (C) wordt door de optelling beïnvloed, evenals de conditie-code (CC), de inter-digit-carry (IDC) en de overflow (OVF). Deze instructie is vooral van belang als tijdens de uitvoering van het programma een constante waarde bij een register geteld dient te worden. In dat geval bespaart men geheugenruimte bij 1-byte operaties, omdat er geen verdere adressering nodig is.

De operatiecodes luiden:

register R0	H'84'
register R1	H'85'
register R2	H'86'
register R3	H'87'

Binaire Code



Executietijd: 3 cycli (9 klokperioden) bij directe adressering
5 cycli (15 klokperioden) bij indirecte adressering

Bits van het PSW die beïnvloed worden: C, CC, IDC, OVF

Beïnvloeding Conditie-Code:

register r	CC1	CC0
positief	0	1
nul	0	0
negatief	1	0

Directe adressering:

I=0
IAR ← IAR+2
ADR ← IAR+adres
R[r] ← R[r]+MEM[ADR]

Indirecte adressering:

I=1
IAR ← IAR+2
ADR ← MEM[IAR+adres]
R[r] ← R[r]+MEM[ADR]

Beschrijving

Deze optelinstructie ter grootte van 2 bytes kan, hetzij in een beperkt geheugengebied (maximaal 128 bytes), dan wel d.m.v. indirecte adressering een byte uit het geheugen optellen bij de inhoud van het aangegeven register. Het resultaat vervangt de inhoud van het desbetreffende register.

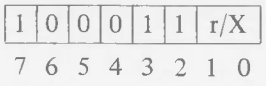
Evenals bij LOAD en STORE RELATIVE, is ook de indirecte wijze van adresseren vooral van groot belang als het hier een variabele betreft. Het indirecte adres kan, bij veelvuldig voorkomen van de variabele, ruimtebesparing betekenen in het geheugen van de microprocessor.

De operatiecodes luiden:

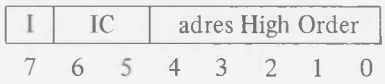
register R0	H'88'
register R1	H'89'
register R2	H'8A'
register R3	H'8B'

Binaire Code

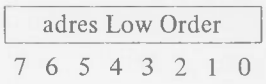
Byte 0



Byte 1



Byte 2



Executietijd: 4 cycli (12 klokperioden) bij directe adressering
6 cycli (18 klokperioden) bij indirecte adressering

PSW-bits die beïnvloed worden: C, CC, IDC, OVF

Beïnvloeding Conditie-Code:

register r	CC1	CC0
positief	0	1
nul	0	0
negatief	1	0

Directe adressering:

```
I=0   IC=00
   IAR ← IAR+3
   ADR ← PAG, adres
   R[r] ← R[r]+MEM[ADR]

I=0   IC=11
   IAR ← IAR+3
   ADR ← (PAG, adres)+R[r]
   R[0] ← R[0]+MEM[ADR]

I=0   IC=01
   IAR ← IAR+3
   R[r] ← R[r]+1
   ADR ← (PAG, adres)+R[r]
   R[0] ← R[0]+MEM[ADR]

I=0   IC=10
   IAR ← IAR+3
   R[r] ← R[r]−1
   ADR ← (PAG, adres)+R[r]
   R[0] ← R[0]+MEM[ADR]
```

Indirecte adressering:

```
I=1   IC=00
   IAR ← IAR+3
   ADR ← MEM[PAG, adres]
   R[r] ← R[r]+MEM[ADR]

I=1   IC=11
   IAR ← IAR+3
   ADR ← MEM[PAG, adres]+R[r]
   R[0] ← R[0]+MEM[ADR]

I=1   IC=01
   IAR ← IAR+3
   R[r] ← R[r]+1
   ADR ← MEM[PAG, adres]+R[r]
   R[0] ← R[0]+MEM[ADR]

I=1   IC=10
   IAR ← IAR+3
   R[r] ← R[r]−1
   ADR ← MEM[PAG, adres]+R[r]
   R[0] ← R[0]+MEM[ADR]
```

Beschrijving

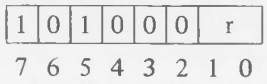
Wordt er niet geïndexeerd, dan zal deze instructie de inhoud van een geheugenplaats optellen bij de inhoud van een gegeven register; het resultaat wordt in hetzelfde register geplaatst. In geval van indexering heeft dit betrekking op register 0 en zal het aangegeven register de index bevatten. Als de IC-waarde 01 of 10 is, dan zal deze index vóór het indexeren met 1 verhoogd, resp. met 1 verlaagd worden. Dit geeft men in de mnemonic aan met (±). Als er geen indirecte adressering plaats heeft, dan is het basisadres gegeven in het adresdeel van byte 1 en byte 2. Bij indirecte adressering treft men, op dit adres (op dezelfde pagina als de instructie), het adres aan van de eigenlijke data. Bij indexering moet deze data met de inhoud van het indexregister verhoogd worden ter bepaling van het effectieve adres. Het effectieve adres bij indexering wordt gevonden door het optellen in zuivere binaire code, d.w.z. dat het effectieve adres wordt bepaald door een basisadres plus de inhoud van het indexregister, die kan variëren van 0 t/m 255.

De operatiecodes luiden:

register R0	H'8C'
register R1	H'8D'
register R2	H'8E'
register R3	H'8F'

Maak de vraagstukken 23 en 24!

Binaire Code



Executietijd: 2 cycli (6 klokperioden)

Bits van het PSW die beïnvloed worden: C, CC, IDC, OVF

Beïnvloeding Conditie-Code:

register 0	CC1	CC0
positief	0	1
nul	0	0
negatief	1	0

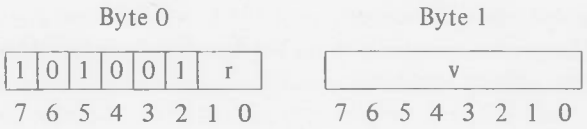
IAR ← IAR+1
R[0] ← R[0]–R[r]

Beschrijving
Deze instructie draagt er zorg voor dat de inhoud van het register r wordt afgetrokken van het register R0. Het resultaat wordt in R0 geplaatst. De aftrekking is in het 2-complement talstelsel.

De operatiecodes luiden:

register R0	H'A0'
register R1	H'A1'
register R2	H'A2'
register R3	H'A3'

Binaire Code



Executietijd: 2 cycli (6 klokperioden)

PSW-bits die worden beïnvloed: C, CC, IDC en OVF

Beïnvloeding Conditie-Code:

register r	CC1	CC0
positief	0	1
nul	0	0
negatief	1	0

IAR ← IAR+2
R[r] ← R[r]–v

Beschrijving

Deze instructie ter grootte van 2 bytes bevat als tweede byte een getal in het 2-complement stelsel. Het wordt van het vermelde register afgetrokken en het resultaat wordt in het register geplaatst. De aftrekking geschiedt in het 2-complement talstelsel. Worden 2 negatieve getallen van elkaar afgetrokken, dan kan het resultaat geen "overflow" geven. De instructie is vooral nuttig als constanten tussen –128 en +127 moeten worden afgetrokken, daar deze instructie geen extra ruimte vergt voor de adressering van de variabele.

De operatiecodes luiden:

register R0	H′A4′
register R1	H′A5′
register R2	H′A6′
register R3	H′A7′

Binaire Code

Executietijd: 3 cycli (9 klokperioden) bij directe adressering
5 cycli (15 klokperioden) bij indirecte adressering

PSW-bits die beïnvloed worden: C, CC, IDC, OVF

Beïnvloeding Conditie-Code:

register r	CC1	CC0
positief	0	1
nul	0	0
negatief	1	0

Directe adressering:

I=0
IAR ← IAR+2
ADR ← IAR+adres
R[r] ← R[r]–MEM[ADR]

Indirecte adressering:

I=1
IAR ← IAR+2
ADR ← MEM[IAR+adres]
R[r] ← R[r]–MEM[ADR]

Beschrijving

Deze instructie ter grootte van 2 bytes kan hetzij in een beperkt geheugengebied (maximaal 128 bytes), dan wel d.m.v. indirecte adressering een byte uit het geheugen aftrekken van de inhoud van het aangegeven register. De aftrekking geschiedt in het 2-complement talstelsel. Het resultaat vervangt de inhoud van het desbetreffende register. Evenals bij ADD RELATIVE (ADDR), is ook de indirecte wijze van adresseren van vooral groot belang als het een variabele betreft. Het indirecte adres kan, bij veelvuldig voorkomen van de variabele, ruimtebesparing betekenen in het geheugen van de microprocessor.

De operatiecodes luiden:

register R0	H′ A8′
register R1	H′ A9′
register R2	H′ AA′
register R3	H′ AB′

Binaire Code

Byte 0

1	0	1	0	1	1	r/X
7	6	5	4	3	2	1 0

Byte 1

I	IC	adres High Order
7	6 5	4 3 2 1 0

Byte 2

adres Low Order
7 6 5 4 3 2 1 0

Executietijd: 4 cycli (12 klokperioden) bij directe adressering
6 cycli (18 klokperioden) bij indirecte adressering

PSW-bits die worden beïnvloed: C, CC, IDC, OVF

Beïnvloeding Conditie-Code:

register r	CC1	CC0
positief	0	1
nul	0	0
negatief	1	0

Directe adressering:

```
I=0   IC=00
   IAR ← IAR+3
   ADR ← PAG, adres
   R[r] ← R[r]–MEM[ADR]

I=0   IC=11
   IAR ← IAR+3
   ADR ← (PAG, adres)+R[r]
   R[0] ← R[0]–MEM[ADR]

I=0   IC=01
   IAR ← IAR+3
   R[r] ← R[r]+1
   ADR ← (PAG, adres)+R[r]
   R[0] ← R[0]–MEM[ADR]

I=0   IC=10
   IAR ← IAR+3
   R[r] ← R[r]–1
   ADR ← (PAG, adres)+R[r]
   R[0] ← R[0]–MEM[ADR]
```

Indirecte adressering:

```
I=1   IC=00
   IAR ← IAR+3
   ADR ← MEM[PAG, adres]
   R[r] ← R[r]–MEM[ADR]

I=1   IC=11
   IAR ← IAR+3
   ADR ← MEM[PAG, adres]+R[r]
   R[0] ← R[0]–MEM[ADR]

I=1   IC=01
   IAR ← IAR+3
   R[r] ← R[r]+1
   ADR ← MEM[PAG, adres]+R[r]
   R[0] ← R[0]–MEM[ADR]

I=1   IC=10
   IAR ← IAR+3
   R[r] ← R[r]–1
   ADR ← MEM[PAG, adres]+R[r]
   R[0] ← R[0]–MEM[ADR]
```

Beschrijving

Deze instructie ter grootte van 3 bytes zal, indien er niet geïndexeerd wordt, de inhoud van een geheugencel aftrekken van de inhoud van een gegeven register; het resultaat wordt dan in hetzelfde register geplaatst. Bij indexering heeft dit betrekking op register 0, en zal het aangegeven register de index bevatten. Afhankelijk van de waarde van IC (01 of 10), wordt deze index vóór het indexeren met 1 verhoogd, resp. met 1 verlaagd. Dit geeft men in de mnemonic aan met (±). Heeft er geen indirecte adressering plaats, dan is het basisadres gegeven in het adresdeel van byte 1 en byte 2. Bij indirecte adressering treft men op dit adres (op dezelfde pagina als de instructie) het adres aan van de eigenlijke data. In het geval van indexering moet deze data met de inhoud van het indexregister worden verhoogd teneinde het effectieve adres te bepalen. Het effectieve adres bij indexering wordt gevonden door optelling in zuivere binaire code, d.w.z. het effectieve adres wordt bepaald door een basisadres plus de inhoud van het indexregister, die kan variëren van 0 t/m 255.

De operatiecodes luiden:

register R0	H' AC'
register R1	H' AD'
register R2	H' AE'
register R3	H' AF'

De microprocessor 2650 kent drie *logische operaties,* namelijk de EN, de OF en de EXCLUSIVE OR (EXOR). Behalve voor normale logische bewerkingen van data, zijn deze operaties bijzonder nuttig voor het aanwijzen van onderdelen van poorten (zie "HARDWARE HANDLEIDING"), hetzij voor de invoer, dan wel voor de uitvoer van data. De 8 pennen van een poort mogen onafhankelijk van elkaar worden gebruikt als in- of uitvoerpennen. Zij worden echter als "groep" toegesproken. Het daarvoor nodige kan heel effectief gebruikt worden m.b.v. bovengenoemde instructies. Bij de in- en uitvoerinstructies wordt op dit deel van de logische instructies uitvoerig teruggekomen. Hier zullen we ons uitsluitend beperken tot logische operaties zoals die door de Boole Algebra zijn gegeven.

Bij de logische operaties moet men bedenken dat een logische operatie uitgevoerd wordt tussen twee operanden. De operanden moeten daarbij niet als één geheel worden beschouwd, maar als een verzameling van 8 elementen. De operatie wordt paarsgewijze op de elementen uitgevoerd, zoals hieronder is vermeld.

RES[i] ← OPAND1[i] ● OPAND2[i]
(● is een operatiesymbool)

Deze operatie geldt voor alle elementen i (bits 0 t/m 7) van een byte. Uit zuinigheidsoverwegingen kan het noodzakelijk zijn dat èlke bit een variabele voorstelt; dat kan ook functioneel wel vereist zijn. Dit laatste is o.a. het geval bij de in- en uitvoer van data; zoals vermeld, wordt hierop in een apart hoofdstuk teruggekomen. Indien de logische operaties echter uitsluitend betrekking hebben op twee variabelen, dan is het eenvoudiger de gehele byte voor deze variabelen te gebruiken en alle elementen identiek te nemen, d.w.z. alle 0 òf alle 1. In dit geval kan ook de Conditie-Code eenvoudig een ondubbelzinnige betekenis hebben, zoals zal blijken. Er is een methodiek om de Conditie-Code te toetsen en deze waarde te gebruiken als logische waarde voor het laten uitvoeren van een sprongopdracht.

Het uitwerken van een Boole-formule kan men eenvoudig realiseren door b.v. de data m.b.v. EN schakelingen te ontwikkelen op het register 0. Men kan eveneens een logisch produkt vormen en het resultaat opslaan in een register zodra er een OF-operatie verschijnt. Men brengt dan het resultaat van het register 0 naar een van de 3 hulpregisters 1 t/m 3 en vervolgens laadt men het register met de eerstvolgende term van een EN-operatie. De volgende variabelen van de EN-operatie worden nu alle weer met deze waarde als operand uitgevoerd. Als dit is geschied en men ontmoet wederom een OF-teken, dan kan men eenvoudig met het vorige register een OF plegen op het register 0 en het resultaat weer terugschrijven in het desbetreffende register. Op deze wijze doorgaand kan men zeer eenvoudig een logische formule berekenen.

Een voorbeeld treffen we hieronder aan:

X ← A.B.C+A.D.+D.F

Nemen we aan dat deze namen alle labels zijn, dan kan het programma er in mnemonics als volgt uitzien:

```
                    commentaar
LODR,0    A         laad A in R0
ANDR,0    B         vorm A.B
ANDR,0    C         vorm A.B.C.
STRZ      1         berg op in R1
LODR,0    A         laad A in R0
ANDR,0    D         vorm A.D
IORZ      1         vorm (A.D) + (A.B.C)
STRZ      1         berg op in R1
LODR,0    D         laad D
ANDR,0    F         vorm D.F
IORZ      1         vorm (D.F) + ((A.D) + (A.B.C))
STRR,0    X         berg resultaat (X) op
```

Na deze korte inleiding zal thans worden overgegaan tot de gedetailleerde behandeling van de "logische" instructies. Thans volgen de WAARHEIDS-tabellen voor A ● B, waarbij de ● een logische EN, OF, of EXCLUSIVE OR operatie voorstelt.

EN:

A bit (0-7)	B bit (0-7)	Resultaat
0	0	0
0	1	0
1	0	0
1	1	1

Deze operatie is bijzonder handig voor z.g. "maskering". Stel men wil de bits 4 t/m 7 van een byte gelijk aan 0 maken en de bits 0 t/m 3 onveranderd laten:

```
byte        10101010
masker      00001111
            --------  EN
resultaat   00001010
```

Logisch EN betekent dus: het resultaat is slechts dan een 1 als zowel bit A als bit B een 1 zijn, anders is het resultaat altijd een 0.

OF:

A bit (0-7)	B bit (0-7)	Resultaat
0	0	0
0	1	1
1	0	1
1	1	1

Deze operatie is eveneens bijzonder handig voor z.g. "maskering". Stel men wil de bits 4 t/m 7 van een byte gelijk aan 1 maken en de bits 0 t/m 3 onveranderd laten:

```
byte        10101010
masker      11110000
            --------  EN
resultaat   11111010
```

Logisch OF betekent dus: het resultaat is slechts dan een 0 als zowel bit A als bit B een 0 zijn, anders is het resultaat altijd een 1.

EXCLUSIVE OR:

A bit (0-7)	B bit (0-7)	Resultaat
0	0	0
0	1	1
1	0	1
1	1	0

Deze operatie komt heel goed van pas als men de bits van een byte wil "inverteren".

Stel men wil de bits 0 t/m 4 van een byte inverteren en de bits 5 t/m 7 onveranderd laten:

```
byte        101|01010
masker      000|11111
            ---+-----  EXCLUSIVE OR
resultaat   101|10101
```

Logisch EXCLUSIVE OR betekent dus: het resultaat is slechts dan een 1 als de bits A en B verschillend zijn, anders is het resultaat altijd een 0.

De *Conditie-Code* geeft bij gemengde I/O patronen van een byte in feite geen informatie over het resultaat. Bestaat het resultaat uitsluitend uit enen of uitsluitend uit nullen, dan zal de Conditie-Code (CC) 10, resp. 00 zijn. Dat zal het geval zijn als de gehele operand slechts één variabele voorstelt. Is er een gemengd patroon van nullen en enen, b.v. als de byte gebruikt wordt voor verschillende variabelen, zoals dit bij in- en uitvoerinstructies kan voorkomen, dan speelt de meest significante bit een overwegende rol. Men moet dan uiterst voorzichtig zijn met de interpretatie van de Conditie-Code, daar die in wezen in sterke mate beïnvloed wordt door de meest significante bit. Deze opmerking geldt voor alle logische instructies; de lezer zal er in voorkomende gevallen op attent gemaakt worden door een opmerking als:

"de CC biedt onzekerheden".

Binaire Code

0	1	0	0	0	0	r
7	6	5	4	3	2	1 0

Executietijd: 2 cycli (6 klokperioden)

PSW-bits die beïnvloed worden: CC

Beïnvloeding Conditie-Code:

register 0	CC1	CC2
positief	0	1
nul	0	0
negatief	1	0

IAR ← IAR+1
R[0] ← R[0]●R[r]

Beschrijving

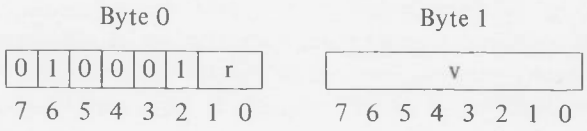
Deze instructie ter grootte van 1 byte zorgt ervoor dat een logische EN wordt verricht tussen de inhoud van een gespecificeerd register r en de inhoud van register 0. Het resultaat vervangt de oorspronkelijke inhoud van het register 0; de inhoud van het register r blijft onveranderd. De Conditie-Code hangt in sterke mate af van de meest significante bit. De CC wordt gezet alsof het resultaat een 2-complement getal is.

De operatiecodes luiden:

register R1	H′41′
register R2	H′42′
register R3	H′43′

De operatiecode H′40′ is gereserveerd voor de HALT-instructie.

Binaire Code



Executietijd: 2 cycli (6 klokperioden)

PSW-bits die beïnvloed worden: CC

Beïnvloeding Conditie-Code:

register 0	CC1	CC0
positief	0	1
nul	0	0
negatief	1	0

IAR ← IAR+2
R[r] ← R[r]●v

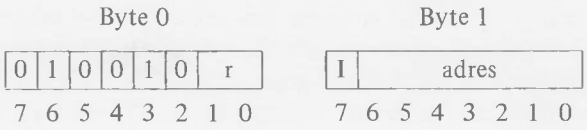
Beschrijving
Deze instructie ter grootte van 2 bytes verricht een logische EN tussen de inhoud, gegeven door byte 1, en de inhoud van het gespecificeerde register. De inhoud van dit register gaat verloren en wordt vervangen door het resultaat. De EN-schakelfunctie is zoals reeds beschreven. De CC wordt gezet alsof het resultaat een 2-complement getal is. Men dient de Conditie-Code voorzichtig te interpreteren.

Opmerking:
Deze instructie kan heel goed dienst doen om bepaalde bits in de 0-stand te plaatsen. Dit is vooral van belang voor de I/O, waar men selectief bepaalde bits in de 0-stand kan plaatsen door deze bits ook 0 te maken in "v".

De operatiecodes luiden:
register R0 H'44'
register R1 H'45'
register R2 H'46'
register R3 H'47'

Maak vraagstuk 25!

Binaire Code



Executietijd: 3 cycli (9 klokperioden) bij directe adressering
5 cycli (15 klokperioden) bij indirecte adressering

PSW-bits die beïnvloed worden: CC

Beïnvloeding Conditie-Code:

register 0	CC1	CC0
positief	0	1
nul	0	0
negatief	1	0

Directe adressering:

I=0
IAR ← IAR+2
ADR ← IAR+adres
R[r] ← R[r]•MEM[ADR]

Indirecte adressering:

I=1
IAR ← IAR+2
ADR ← MEM[IAR+adres]
R[r] ← R[r]•MEM[ADR]

Beschrijving

Deze instructie ter grootte van 2 bytes veroorzaakt een logische EN-operatie tussen de inhoud van het gespecificeerde register r en de data, aanwezig op het effectieve adres. Het resultaat vervangt de inhoud van het register r. Bovendien beïnvloedt het resultaat de stand van het Conditie-Code register, waarbij de nieuwe inhoud van het register r wordt geïnterpreteerd als een 2-complement getal. Uiterste voorzichtigheid bij de interpretatie is dus geboden.

De operatiecodes luiden:

register R0	H'48'
register R1	H'49'
register R2	H'4A'
register R3	H'4B'

Binaire Code

Byte 0

0	1	0	0	1	1	r/X
7	6	5	4	3	2	1 0

Byte 1

I	IC	adres High Order
7	6 5	4 3 2 1 0

Byte 2

adres Low Order
7 6 5 4 3 2 1 0

Executietijd: 4 cycli (12 klokperioden) bij directe adressering
6 cycli (18 klokperioden) bij indirecte adressering

PSW-bits die beïnvloed worden: CC

Beïnvloeding Conditie-Code:

register 0	CC1	CC0
positief	0	1
nul	0	0
negatief	1	0

Directe adressering:

```
I=0   IC=00
   IAR ← IAR+3
   ADR ← PAG, adres
   R[r] ← R[r]•MEM[ADR]

I=0   IC=11
   IAR ← IAR+3
   ADR ← (PAG, adres)+R[r]
   R[0] ← R[0]•MEM[ADR]

I=0   IC=01
   IAR ← IAR+3
   R[r] ← R[r]+1
   ADR ← (PAG, adres)+R[r]
   R[0] ← R[0]•MEM[ADR]

I=0   IC=10
   IAR ← IAR+3
   R[r] ← R[r]−1
   ADR ← (PAG, adres)+R[r]
   R[0] ← R[0]•MEM[ADR]
```

Indirecte adressering:

```
I=1   IC=00
   IAR ← IAR+3
   ADR ← MEM[PAG, adres]
   R[r] ← R[r]•MEM[ADR]

I=1   IC=11
   IAR ← IAR+3
   ADR ← MEM[PAG, adres]+R[r]
   R[0] ← R[0]•MEM[ADR]

I=1   IC=01
   IAR ← IAR+3
   R[r] ← R[r]+1
   ADR ← MEM[PAG, adres]+R[r]
   R[0] ← R[0]•MEM[ADR]

I=0   IC=10
   IAR ← IAR+3
   R[r] ← R[r]−1
   ADR ← MEM[PAG, adres]+R[r]
   R[0] ← R[0]•MEM[ADR]
```

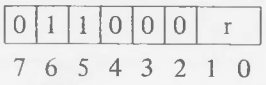
Beschrijving

Deze instructie ter grootte van 3 bytes bewerkstelligt, indien er niet geïndexeerd wordt, dat de logische EN-operatie verricht wordt tussen de inhoud van een geheugenplaats en de inhoud van een gegeven register. Het resultaat wordt in hetzelfde register geplaatst. In geval van indexering heeft dit betrekking op register 0 en zal het aangegeven register de index bevatten. Afhankelijk van de waarde IC (01 of 10), wordt deze index vóór het indexeren met 1 verhoogd, resp. met 1 verlaagd. Dit geeft men in de mnemonic aan met (±). Als er geen indirecte adressering plaats heeft, dan is het basisadres gegeven in het adresdeel van byte 1 en byte 2. Bij indirecte adressering treft men op dit adres (op dezelfde pagina als de instructie) het adres aan van de eigenlijke data. In geval van indexering moet men de inhoud op dit adres met de inhoud van het index-register verhogen om het effectieve adres te bepalen. Het effectieve adres bij indexering wordt gevonden door het optellen in zuivere binaire code, d.w.z. het effectieve adres wordt bepaald door een basisadres plus de inhoud van het index-register, die kan variëren van 0 t/m 255.

De operatiecodes luiden:

register R0	H′4C′
register R1	H′4D′
register R2	H′4E′
register R3	H′4F′

Binaire Code



Executietijd: 2 cycli (6 klokperioden)

PSW-bits die beïnvloed worden: CC

Beïnvloeding Conditie-Code:

register 0	CC1	CC0
positief	0	1
nul	0	0
negatief	1	0

IAR ← IAR+1
R[0] ← R[0]●R[r]

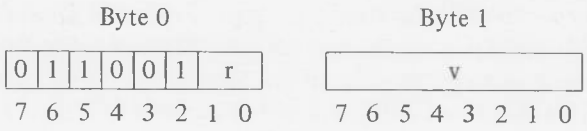
Beschrijving

Deze instructie ter grootte van 1 byte bewerkstelligt dat de logische OF verricht wordt tussen de inhoud van register 0 en het aangegeven register. Het resultaat wordt in register 0 geplaatst. Ondergaan 2 vectoren de logische operatie, dan zal de inhoud van de Conditie-Code weer sterk afhankelijk zijn van de aard van de vectoren. Het resultaat wordt opgevat als een 2-complement getal.

De operatiecodes luiden:

register R0	H'60'
register R1	H'61'
register R2	H'62'
register R3	H'63'

Binaire Code



Executietijd: 2 cycli (6 klokperioden)

PSW-bits die beïnvloed worden: CC

Beïnvloeding Conditie-Code:

register r	CC1	CC0
positief	0	1
nul	0	0
negatief	1	0

IAR ← IAR+2
R[r] ← R[r]●v

Beschrijving
Deze instructie ter grootte van 2 bytes verricht een logische OF tussen de inhoud van het register r en de in de instructie bijgevoegde byte 1. Het resultaat vervangt de oorspronkelijke inhoud van het register r. Uiterste zorgvuldigheid is weer geboden met de interpretatie van de Conditie-Code.

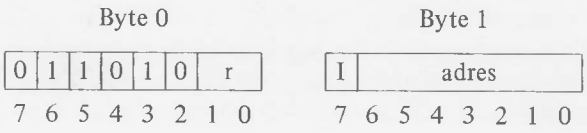
Opmerking:
Van deze instructie maakt men met voordeel gebruik voor het in de 1-stand plaatsen van bepaalde bits. Dit is vooral van belang voor de I/O.

De operatiecodes luiden:

register R0	H'64'
register R1	H'65'
register R2	H'66'
register R3	H'67'

Maak vraagstuk 26!

Binaire Code



Executietijd: 3 cycli (9 kolokperioden) bij directe adressering
5 cycli (15 klokperioden) bij indirecte adressering

PSW-bits die beïnvloed worden: CC

Beïnvloeding Conditie-Code:

register r	CC1	CC0
positief	0	1
nul	0	0
negatief	1	0

Directe adressering:

```
I=0
   IAR ← IAR+2
   ADR ← IAR+adres
   R[r] ← R[r]•MEM[ADR]
```

Indirecte adressering:

```
I=1
   IAR ← IAR+2
   ADR ← MEM[IAR+adres]
   R[r] ← R[r]•MEM[ADR]
```

Beschrijving

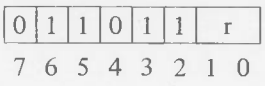
Deze instructie ter grootte van 2 bytes veroorzaakt een logische OF-operatie tussen de inhoud van het gespecificeerde register r en de data op het effectieve adres. Het resultaat vervangt de inhoud van het desbetreffende register. De Conditie-Code wordt ook hier beïnvloed door de inhoud van het register na afloop van het uitvoeren van de instructie; wederom is uiterste voorzichtigheid met de interpretatie van de Conditie-Code dan ook geboden.

De operatiecodes luiden:

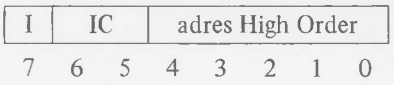
register R0	H'68'
register R1	H'69'
register R2	H'6A'
register R3	H'6B'

Binaire Code

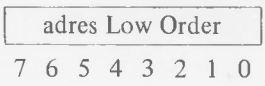
Byte 0



Byte 1



Byte 2



Executietijd: 4 cycli (12 klokperioden) bij directe adressering
6 cycli (18 klokperioden) bij indirecte adressering

PSW-bits die beïnvloed worden: CC

Beïnvloeding Conditie-Code:

register 0	CC1	CC0
positief	0	1
nul	0	0
negatief	1	0

Directe adressering:

```
I=0   IC=00
   IAR ← IAR+3
   ADR ← PAG, adres
   R[r] ← R[r]•MEM[ADR]

I=0   IC=11
   IAR ← IAR+3
   ADR ← (PAG, adres)+R[r]
   R[0] ← R[0]•MEM[ADR]

I=0   IC=01
   IAR ← IAR+3
   R[r] ← R[r]+1
   ADR ← (PAG, adres)+R[r]
   R[0] ← R[0]•MEM[ADR]

I=0   IC=10
   IAR ← IAR+3
   R[r] ← R[r]−1
   ADR ← (PAG, adres)+R[r]
   R[0] ← R[0]•MEM[ADR]
```

Indirecte adressering:

```
I=1   IC=00
   IAR ← IAR+3
   ADR ← MEM[PAG, adres]
   R[r] ← R[r]•MEM[ADR]

I=1   IC=11
   IAR ← IAR+3
   ADR ← MEM[PAG, adres]+R[r]
   R[0] ← R[0]•MEM[ADR]

I=1   IC=01
   IAR ← IAR+3
   R[r] ← R[r]+1
   ADR ← MEM[PAG, adres]+R[r]
   R[0] ← R[0]•MEM[ADR]

I=1   IC=10
   IAR ← IAR+3
   R[r] ← R[r]−1
   ADR ← MEM[PAG, adres]+R[r]
   R[0] ← R[0]•MEM[ADR]
```

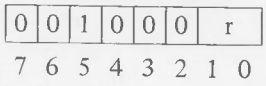
Beschrijving

Deze instructie ter grootte van 3 bytes zal, indien er niet geïndexeerd wordt, een logische OF-operatie verrichten tussen de inhoud van de geheugenplaats en de inhoud van een gegeven register r; het resultaat wordt in hetzelfde register geplaatst. In geval van indexering heeft dit betrekking op register 0 en zal het aangegeven register de index bevatten. Afhankelijk van de waarde van IC (01 of 10), wordt deze index vóór het indexeren met 1 verhoogd, resp. met 1 verlaagd. Dit geeft men in de mnemonic aan met (±). Heeft er geen indirecte adressering plaats, dan is het basisadres gegeven in het deel van byte 1 en byte 2. Bij indirecte adressering treft men op dit adres (op dezelfde pagina als de instructie) het adres aan van de eigenlijke data. Bij indexering moet de inhoud met de inhoud van het index-register verhoogd worden teneinde het effectieve adres te verkrijgen. Het effectieve adres bij indexering wordt gevonden door het optellen in zuivere binaire code, d.w.z. dat het effectieve adres wordt bepaald door een basisadres plus de inhoud van het index-register, die kan variëren van 0 t/m 255.

De operatiecodes luiden:

register R0	H′6C′
register R1	H′6D′
register R2	H′6E′
register R3	H′6F′

Binaire Code



Executietijd: 2 cycli (6 klokperioden)

PSW-bits die beïnvloed worden: CC

Beïnvloeding Conditie-Code:

register 0	CC1	CC0
positief	0	1
nul	0	0
negatief	1	0

IAR ← IAR+1
R[0] ← R[0]●R[r]

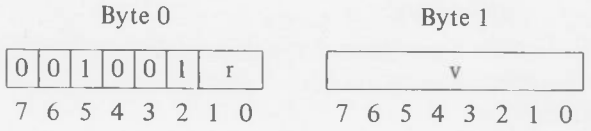
Beschrijving

Deze instructie ter grootte van 1 byte veroorzaakt een logische EXCLUSIVE OR tussen het register 0 en het gespecificeerde register. Het resultaat vervangt de oorspronkelijke inhoud van het register 0. De Conditie-Code wordt daardoor beïnvloed. In verband met de mogelijkheid dat er een vector aanwezig kan zijn, is uiterste voorzichtigheid met de interpretatie van de Conditie-Code weer geboden.

De operatiecodes luiden:

register R0	H'20'
register R1	H'21'
register R2	H'22'
register R3	H'23'

Binaire Code



Executietijd: 2 cycli (6 klokperioden)

PSW-bits die beïnvloed worden: CC

Beïnvloeding Conditie-Code:

register r	CC1	CC0
positief	0	1
nul	0	0
negatief	1	0

IAR ← IAR+2
R[r] ← R[r]●v

Beschrijving
Deze instructie ter grootte van 2 bytes veroorzaakt een logische EXCLUSIVE OR tussen het gespecificeerde register en de in de instructie opgegeven data (byte 1). Het resultaat van deze operatie vervangt de oorspronkelijke inhoud van het register r. De Conditie-Code wordt gezet.

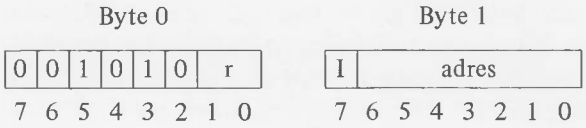
Opmerking:
Deze Immediate Exclusive Or wordt vaak gebruikt om selectief bepaalde bits te inverteren. Op de plaatsen waar een 0 aanwezig is in de betreffende byte, zal geen inversie plaats hebben, terwijl op de plaatsen, waar 1 aanwezig is, de bit in het desbetreffende register wordt geïnverteerd. Dit is ook van belang bij de verwerking van data voor de in- en uitvoer.

De operatiecodes luiden:

register R0	H'24'
register R1	H'25'
register R2	H'26'
register R3	H'27'

Maak nu vraagstuk 27!

Binaire Code



Executietijd: 3 cycli (9 klokperioden) bij directe adressering
5 cycli (15 klokperioden) bij indirecte adressering

PSW-bits die beïnvloed worden: CC

Beïnvloeding Conditie-Code:

register r	CC1	CC0
positief	0	1
nul	0	0
negatief	1	0

Directe adressering:

I=0
IAR ← IAR+2
ADR ← IAR+adres
R[r] ← R[r]•MEM[ADR]

Indirecte adressering:

I=1
IAR ← IAR+2
ADR ← MEM[IAR+adres]
R[r] ← R[r]•MEM[ADR]

Beschrijving

Deze instructie ter grootte van 2 bytes veroorzaakt een logische EXCLUSIVE OR tussen de inhoud van register r en de data die gespecificeerd zijn door het effectieve adres. Het resultaat vervangt de oorspronkelijke inhoud van het register r.

De operatiecodes luiden:

register R0	H'28'
register R1	H'29'
register R2	H'2A'
register R3	H'2B'

Binaire Code

Byte 0

0	0	1	0	1	1	r/X
7	6	5	4	3	2	1 0

Byte 1

I	IC	adres High Order
7	6 5	4 3 2 1 0

Byte 2

adres Low Order
7 6 5 4 3 2 1 0

Executietijd: 4 cycli (12 klokperioden) bij directe adressering
6 cycli (18 klokperioden) bij indirecte adressering

PSW-bits die beïnvloed worden: CC

Beïnvloeding Conditie-Code:

register r	CC1	CC0
positief	0	1
nul	0	0
negatief	1	0

Directe adressering:

I=0 IC=00
IAR ← IAR+3
ADR ← PAG, adres
R[r] ← R[r]•MEM[ADR]

I=0 IC=11
IAR ← IAR+3
ADR ← (PAG, adres)+R[r]
R[0] ← R[0]•MEM[ADR]

I=0 IC=01
IAR ← IAR+3
R[r] ← R[r]+1
ADR ← (PAG, adres)+R[r]
R[0] ← R[0]•MEM[ADR]

I=0 IC=10
IAR ← IAR+3
R[r] ← R[r]−1
ADR ← (PAG, adres)+R[r]
R[0] ← R[0]•MEM[ADR]

Indirecte adressering:

I=1 IC=00
IAR ← IAR+3
ADR ← MEM[PAG, adres]
R[r] ← R[r]•MEM[ADR]

I=1 IC=11
IAR ← IAR+3
ADR ← MEM[PAG, adres]+R[r]
R[0] ← R[0]•MEM[ADR]

I=1 IC=01
IAR ← IAR+3
R[r] ← R[r]+1
ADR ← MEM[PAG, adres]+R[r]
R[0] ← R[0]•MEM[ADR]

I=1 IC=10
IAR ← IAR+3
R[r] ← R[r]−1
ADR ← MEM[PAG, adres]+R[r]
R[0] ← R[0]•MEM[ADR]

Beschrijving

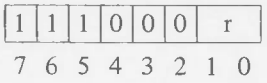
Deze instructie ter grootte van 3 bytes zal, indien er niet geïndexeerd wordt, een logische EXCLUSIVE OR-operatie verrichten tussen de inhoud van een geheugenplaats en de inhoud van een gegeven register r. Het resultaat wordt in hetzelfde register geplaatst. Bij indexering heeft dit betrekking op register 0 en zal het aangegeven register de index bevatten. Afhankelijk van de IC-waarde (01 of 10), wordt deze index vóór het indexeren met 1 verhoogd, resp. met 1 verlaagd. Dit geeft men in de mnemonic aan met (±). Heeft geen indirecte adressering plaats, dan is het basisadres gegeven in het adresdeel van byte 1 en byte 2. Bij indirecte adressering treft men op dit adres (op dezelfde pagina als de instructie) het adres aan van de eigenlijke data. Bij indexeren moet dit adres met de inhoud van het indexregister verhoogd worden. Het effectieve adres bij indexering wordt gevonden door optellen in zuivere binaire code, d.w.z. het effectieve adres wordt bepaald door een basisadres plus de inhoud van het index-register, die kan variëren van 0 t/m 255.

De operatiecodes luiden:

register R0	H′ 2C′
register R1	H′ 2D′
register R2	H′ 2E′
register R3	H′ 2F′

Bij het vergelijken van getallen en vectoren maakt de microprocessor 2650 onderscheid tussen getallen en vectoren. Bij het vergelijken van getallen is de microprocessor actief in de 2-complement code. Op grond van deze code wordt het Conditie-Code register in een bepaalde stand geplaatst. Voor dit doel moet het PSW-bit COM 0 zijn (de COM bit wordt beïnvloed door een instructie die later behandeld zal worden). Indien de COM bit 1 is, zal er een logische vergelijking van beide inhouden plaats hebben. Dit houdt in dat de beide getallen worden opgevat als twee 8-bits getallen, d.w.z. ze zijn 0 of groter dan 0 en dus maximaal 255 (dus binaire getallen zonder teken en altijd positief).

Binaire Code



Executietijd: 2 cycli (6 klokperioden)

PSW-bits die beïnvloed worden: CC

Beïnvloeding Conditie-Code:	CC1	CC0
reg. 0 groter dan reg. r	0	1
reg. 0 gelijk aan reg. r	0	0
reg. 0 kleiner dan reg. r	1	0

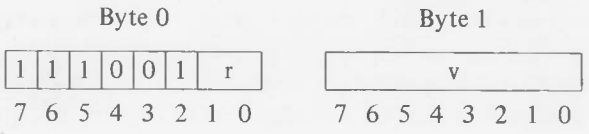
Beschrijving

Deze instructie ter grootte van 1 byte bewerkstelligt dat de inhoud van het gespecificeerde register r wordt vergeleken met de inhoud van het register 0. Geen van beide registers wordt door deze vergelijking beïnvloed. De vergelijking geschiedt hetzij in de "rekenkundige" mode (2-complement), dan wel in de "logische" mode (2-absolute waarden), veroorzaakt door de COM-bit (resp. 0 of 1).

De operatiecodes luiden:

register R0	H′ E0′
register R1	H′ E1′
register R2	H′ E2′
register R3	H′ E3′

Binaire Code



Executietijd: 3 cycli (9 klokperioden)

PSW-bits die beïnvloed worden: CC

Beïnvloeding Conditie-Code:	CC1	CC0
reg. 0 groter dan v	0	1
reg. 0 gelijk aan v	0	0
reg. 0 kleiner dan v	1	0

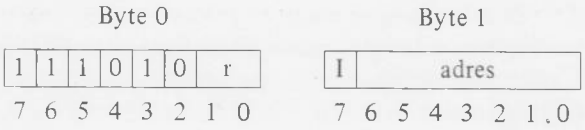
Beschrijving

Deze instructie ter grootte van 2 bytes bewerkstelligt de vergelijking van de inhoud van het gespecificeerde register r met de inhoud van de tweede byte van deze instructie. Deze vergelijking wordt hetzij rekenkundig dan wel logisch uitgevoerd, afhankelijk van de waarde van de COM-bit van het PSW. Is de COM-bit 1, dan zal de waarde van het resultaat worden beschouwd als een 8-bits positief binair getal, en als de COM-bit 0 is, dan zal deze waarde worden beschouwd als een 2-complement getal. De inhoud van het register wordt niet beïnvloed.

De operatiecodes luiden:

register R0	H′ E4′
register R1	H′ E5′
register R2	H′ E6′
register R3	H′ E7′

Binaire Code



Executietijd: 3 cycli (9 klokperioden) bij directe adressering
5 cycli (15 klokperioden) bij indirecte adressering

PSW-bits die beïnvloed worden: CC

Beïnvloeding Conditie-Code:	CC1	CC0
reg. 0 > geheugenbyte	0	1
reg. 0 = geheugenbyte	0	0
reg. 0 < geheugenbyte	1	0

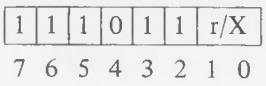
Beschrijving
Deze instructie ter grootte van 2 bytes vergelijkt de inhoud van een gespecificeerd register r met de inhoud die op het effectieve adres aanwezig is. De vergelijking wordt uitgevoerd hetzij in rekenkundige, dan wel in logische mode, afhankelijk of de COM-bit 0 resp. 1 is. Als de COM-bit 1 is (logische mode), dan zal het resultaat behandeld worden als een 8-bits positief binair getal; is de COM-bit daarentegen 0, dan zal het resultaat behandeld worden als een 8-bits 2-complement getal.

De operatiecodes luiden:

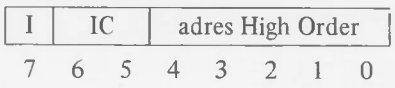
register R0	H'E8'
register R1	H'E9'
register R2	H'EA'
register R3	H'EB'

Binaire Code

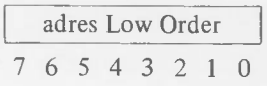
Byte 0



Byte 1



Byte 2



Executietijd: 4 cycli (12 klokperioden) bij directe adressering
6 cycli (18 klokperioden) bij indirecte adressering

PSW-bits die beïnvloed worden: CC

Beïnvloeding Conditie-Code:	CC1	CC0
reg. r > geheugenbyte	0	1
reg. r = geheugenbyte	0	0
reg. r < geheugenbyte	1	0

Beschrijving

Deze instructie zal, indien er niet geïndexeerd wordt, de inhoud van een geheugenplaats vergelijken met de inhoud van een gegeven register r; het resultaat beïnvloedt de Conditie-Code, maar de inhoud van het register wordt niet aangetast. In geval van indexering heeft dit betrekking op reg. 0 en zal het aangegeven register de index bevatten. Als de IC-waarde 01 of 10 is, dan zal deze index vóór het indexeren met 1 verhoogd, resp. met 1 verlaagd worden. Heeft er geen indirecte adressering plaats, dan is het basisadres gegeven in het adresdeel van byte 1 en byte 2. Bij indirecte adressering treft men op dit adres (op dezelfde pagina als de instructie) het adres aan van de eigenlijke data. Bij indexering moet men de inhoud van het indexregister verhogen om het effectieve adres te bepalen. Het effectieve adres bij indexering wordt gevonden door het optellen in zuivere binaire code, d.w.z. dat het effectieve adres wordt bepaald door een basis-adres plus de inhoud van het indexregister, die kan variëren van 0 t/m 255.

De operatiecodes luiden:

register R0	H'EC'
register R1	H'ED'
register R2	H'EE'
register R3	H'EF'

De microprocessor 2650 kan op alle registers, d.w.z. register 0 en de registers 1 t/m 3 van bank 0 en van bank 1, *rotaties* uitvoeren.

Deze rotaties kunnen betrekking hebben op het gespecificeerde register, resp. op het gespecificeerde register en de CARRY-bit C. Dit laatste hangt af van de stand van de bit WC in het PSW. Als WC = 0, is de CARRY-bit niet betrokken bij de rotatie en als WC = 1, dan is de CARRY-bit wèl betrokken bij de rotatie. Ook de Conditie-Code wordt door de rotatie beïnvloed. De inhoud van het register na de rotatie wordt beschouwd als een getal in het 2-complement binaire talstelsel. Verandert bit 7 van het desbetreffende register van waarde, dan zal de overflow-bit OVF in het PSW in de 1-stand gezet worden. Verandert de inhoud niet, dan is de OVF-bit 0. Ook de Inter Digit Carry (IDC)-bit van het PSW wordt door de rotaties beïnvloed. Dit geldt echter uitsluitend onder de voorwaarde WC = 1.

Binaire Code

1	1	0	1	0	0	r	
7	6	5	4	3	2	1	0

Executietijd: 2 cycli (6 klokperioden)

PSW-bits die beïnvloed worden: C, CC, IDC, OVF

Beïnvloeding Conditie-Code:

register r	CC1	CC0
positief	0	1
nul	0	0
negatief	1	0

Beschrijving

Deze instructie ter grootte van 1 byte verschuift de inhoud van het gespecificeerde register r over 1 plaats naar links. Indien WC = 0, dan zal bit 7 doorgeschoven worden naar bit 0 van het register. Op deze wijze wordt een rotatie verkregen (zie fig. 11).

Voor het geval dat WC = 1, zal bit 7 doorgeschoven worden naar de bit C (de CARRY-bit van het PSW). Deze wordt op zijn beurt verplaatst naar bit 0 van het desbetreffende register. Op deze wijze wordt een rotatie over 9 bits verkregen. Door de CARRY-bit na een rotatie telkens te vervangen d.m.v. een later te behandelen speciale instructie, hetzij een 1, hetzij een 0, kan aan de rechterzijde van het register een serie enen of nullen ingeschoven worden (zie fig. 12).

De rotatie is van groot belang bij het vermenigvuldigen van binaire getallen. Bij een dergelijke vermenigvuldiging dient gebruik gemaakt te worden van een subroutine. Daar bij het vermenigvuldigen een getal doorgaans groter wordt dan 1 byte, zal men op een gegeven ogenblik bit 7 moeten kunnen doorgeven aan de volgende byte, namelijk de meest significante byte van de vermenigvuldiging. Deze koppeling in transport kan worden verkregen door bit 7 in de CARRY-bit te plaatsen en de volgende rotatie uit te voeren op de meer significante byte. Bij deze rotatie wordt de in de CARRY-bit aanwezige bit 7 nu geplaatst in bit 0 van de meer significante byte. Op deze wijze wordt de overdracht van deze bit verzekerd.

De operatiecodes luiden:

register R0	H' D0'
register R1	H' D1'
register R2	H' D2'
register R3	H' D3'

Maak nu vraagstuk 27!

Binaire Code

0	1	0	1	0	0	r	
7	6	5	4	3	2	1	0

Executietijd: 2 cycli (6 klokperioden)

PSW-bits die beïnvloed worden: C, CC, IDC, OVF

Beïnvloeding Conditie-Code:

register r	CC1	CC0
positief	0	1
nul	0	0
negatief	1	0

Beschrijving

Bij een rotatie naar rechts zal een bepaalde bit doorgeschoven worden naar zijn rechterbuurman. Zo zal bit 5 de waarde aannemen van bit 6, bit 4 de waarde van bit 5, enz. Indien WC = 0, zal bit 7 de waarde van bit 0 aannemen; zie fig. 13.

Als WC = 1, zal de waarde van de CARRY-bit C door bit 7 worden aangenomen; deze CARRY-bit zal op zijn beurt de waarde overnemen van bit 0 van de desbetreffende byte; zie fig. 14. De Conditie-Code geeft de inhoud weer van het desbetreffende register na afloop van de verplaatsing. De inhoud wordt opgevat als een getal in het 2-complement binaire talstelsel.

De operatiecodes luiden:

register R0	H′ 50′
register R1	H′ 51′
register R2	H′ 52′
register R3	H′ 53′

Het *Programma-Status Woord* (*PSW*) is een verzameling van flip-flops die speciale toestanden in de microprocessor 2650 registreren, of die op bepaalde acties in de microprocessor reageren, zoals b.v. de Conditie-Code (CC) bits. Tijdens de werking van een machine zullen deze bits herhaaldelijk geraadpleegd moeten worden, resp. in stand 0 of 1 worden gezet.

Er zijn programma's denkbaar waarbij men de inhoud van een Programma-Status Woord in zijn geheel wenst te redden, resp. nadat het gered is, weer in het PSW te laden. Deze speciale communicatie geschiedt via het register R0 en kan uiteraard niet betrekking hebben op meer bits dan het register breed is, namelijk 8 bits. Het PSW bevat echter meer dan 8 bits, zodat het verdeeld is in 2 delen, het PSL (Program Status Word Lower) en het PSU (Program Status Word Upper). De inhouden van deze beide delen kunnen overgedragen worden aan het register R0. Daarna kan men d.m.v. een STORE-instructie de inhoud hetzij in een register plaatsen, dan wel in het Random Access Memory (RAM). Ook de omgekeerde datastroom is mogelijk: men kan data uit het geheugen of uit een van de registers in register R0 plaatsen, en van daar uit in het PSU, resp. PSL. Ook is het mogelijk – nadat een PSU of PSL in het register R0 is geplaatst – logische bewerkingen op het register R0 uit te voeren, waardoor de inhoud van dat register uiteraard wordt gewijzigd. Na deze wijziging kan men de inhoud opnieuw in het PSU of het PSL terugplaatsen. Op deze wijze kan men bits van deze beide delen van het PSW beïnvloeden.

Het is echter niet nodig om deze omslachtige wijze van "store", "bewerk" en "laad", in het totaal dus 3 instructies, uit te voeren om een bit in het PSW te wijzigen. Voor dat doel zijn er twee speciale instructies aanwezig, de z.g. SET- en CLEAR-instructie.

Naast de instructies, die de inhoud van het PSW kunnen wijzigen, zijn er ook instructies die de inhoud van het PSW kunnen onderzoeken. Dit zijn de TEST Program Status Word instructies. Op de volgende bladzijden worden deze instructies nader uiteen gezet.

Binaire Code

0	0	0	1	0	0	1	0
7	6	5	4	3	2	1	0

Executietijd: 2 cycli (6 klokperioden)

PSW-bits die beïnvloed worden: CC

Beïnvloeding Conditie-Code:

register 0	CC1	CC0
positief	0	1
nul	0	0
negatief	1	0

Beschrijving

Deze instructie ter grootte van 1 byte zorgt ervoor dat de inhoud van het PSW Upper (PSU) wordt overgedragen aan register R0. De bits 3 en 4 van het PSU worden als 0 geïnterpreteerd en verschijnen dus ook als 0 in het register R0. De inhoud van het register R0 wordt voor de Conditie-Code beschouwd als een getal in het 2-complement binaire talstelsel. Deze instructie is van bijzonder belang voor het geval dat naar een subroutine wordt gesprongen. Bij het springen naar een subroutine wordt het terugkeeradres op een stapel (Stack) gered. Dit is echter niet het geval met het Program Status Word. De programmeur dient dus zorg te dragen dat het PSW zo nodig gered wordt. Met behulp van deze instructie kan men het PSW Upper (PSU) redden via R0 en tijdelijk hetzij in een register dan wel in een geheugenplaats registreren.

De operatiecode luidt:
H′ 12′

Binaire Code

0	0	0	1	0	0	1	1
7	6	5	4	3	2	1	0

Executietijd: 2 cycli (6 klokperioden)

PSW-bits die beïnvloed worden: CC

Beïnvloeding Conditie-Code:

register 0	CC1	CC0
positief	0	1
nul	0	0
negatief	1	0

Beschrijving

Met behulp van deze instructie kan de inhoud van het PSW Lower (PSL) worden overgedragen aan het register R0. Deze instructie is, evenals bij het STORE PSW Upper, van belang voor het geval dat men gebruik maakt van een subroutine. Als tijdens deze subroutine de CC-, COM-, IDC-, RS-bit e.d. beïnvloed worden, dan kan men m.b.v. deze instructies het totale PSL redden en tijdelijk via R0, hetzij in een register dan wel in het geheugen registreren.

De operatiecode luidt:
H' 13'

Binaire Code

1	0	0	1	0	0	1	0
7	6	5	4	3	2	1	0

Executietijd: 2 cycli (6 klokperioden)

PSW-bits die beïnvloed worden: F, II, SP

Beschrijving

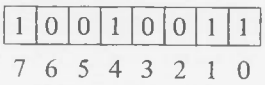
Deze instructie ter grootte van 1 byte zorgt ervoor dat de inhoud van het register R0 wordt overgedragen aan het PSW Upper (PSU). Deze instructie wordt vooral daar gebruikt waar, b.v. in een subroutine, eerst het PSU is gered bij het begin van een subroutine. Aan het einde van de subroutine kan het oorspronkelijk PSW Upper dan via register R0 uit het geheugen of een register hersteld worden.

Men kan de instructie ook gebruiken voor de initialisatie van de processor.

De operatiecode luidt:
H′ 92′

Maak de vraagstukken 29 en 30!

Binaire Code



Executietijd: 2 cycli (6 klokperioden)

PSW-bits die beïnvloed worden: CC, IDC, RS, WC, OVF, COM, C

Beïnvloeding Conditie-Code: de CC krijgt de waarde die in bit 7 en bit 6 van reg. R0 staan.

Beschrijving

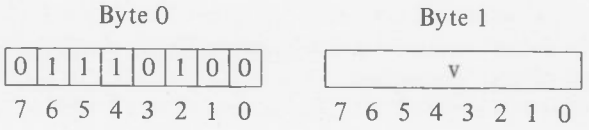
Deze instructie ter grootte van 1 byte zorgt ervoor dat de inhoud van register R0 in het PSW Lower (PSL) wordt geplaatst. Evenals bij de vorige instructie, kan dit van belang zijn b.v. aan het einde van een subroutine. Als het PSW gered is, dan is het nodig aan het einde van de subroutine het PSW weer terug te plaatsen. Mocht tijdens de uitvoering van de subroutine de CC-, OVF-bit e.d. in een bepaalde stand zijn gebracht, dan kan het wenselijk zijn deze waarde te redden. Men dient dit dan te doen via een STORE en b.v. het PSW dan in een register te plaatsen. Pas daarna draagt men de inhoud van het oude PSW over aan het PSW-register.

Men kan deze instructie ook gebruiken bij de initialisatie van de processor.

De operatiecode luidt:
H' 93'

Maak de vraagstukken 31 en 32!

Binaire Code



Executietijd: 3 cycli (9 klokperioden)

PSW-bits die beïnvloed worden: F, II, SP

Beschrijving

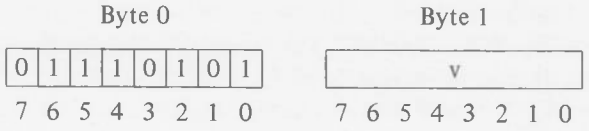
Met behulp van deze instructie is het mogelijk bepaalde bits in het PSW Upper (PSU) te "resetten". Daartoe onderzoekt de microprocessor 2650 de "immediate vector" "v" op enen. Als een 1 wordt aangetroffen, dan wordt de corresponderende bit in het PSU in de 0-stand gezet. Is in de vector een bepaald element 0, dan zal het overeenkomstige element in het PSU niet beïnvloed worden.

Sommige bits van het PSW zijn vaker aan beïnvloeding onderhevig dan andere. Zo zal de "stackpointer" nauwelijks d.m.v. instructies bestuurd worden. Anders is het gesteld met de FLAG-bit en de Interrupt-Inhibit (II)-bit. Voor deze beide bits zal hier de hexadecimale code worden weergegeven. Men kan in het algemeen stellen dat bits die beïnvloed dienen te worden (hetzij "set", dan wel "reset"), aangegeven worden door een 1 in het "masker" "v".

	Hex. Code	Binaire Code
RESET FLAG bit:	H'7440'	01110100 01000000
RESET II bit:	H'7420'	01110100 00100000

De operatiecode luidt:
H'74'

Binaire Code



Executietijd: 3 cycli (9 klokperioden)

PSW-bits die beïnvloed worden: CC, IDC, RS, WC, OVF, COM, C

Beïnvloeding Conditie-Code: de CC-bits kunnen d.m.v. deze instructie 0 gemaakt worden.

Beschrijving

Deze instructie kan een aantal bits in het PSW Lower (PSL) in de 0-stand zetten. De geselecteerde bits worden in het "masker" "v" aangegeven m.b.v. een 1. De instructie is heel geschikt om bepaalde bits stuk voor stuk in het PSL in de 0-stand te plaatsen. Enkele specifieke voorbeelden zijn:

	Hex. Code	Binaire Code
Register Bank Select (RS)	H′7510′	01110101 00010000

Deze instructie zorgt ervoor dat Registerbank 0 wordt geselecteerd.

Gebruik geen Carry: WC = 0	H′7508′	01110101 00001000
Rekenkundig vergelijken: COM = 0	H′7502′	01110101 00000010
Maak CARRY bit 0 : C = 0	H′7501′	01110101 00000001

De operatiecode luidt:
H′75′

Binaire Code

Executietijd: 3 cycli (9 klokperioden)

PSW-bits die beïnvloed worden: F, II, SP

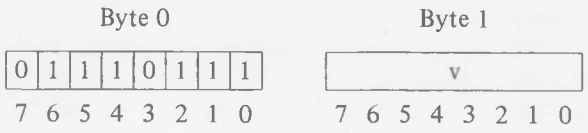
Beschrijving

Deze instructie ter grootte van 2 bytes selecteert m.b.v. enen de plaatsen waarop in het PSW Upper (PSU) de bits in de 1-stand geplaatst dienen te worden. De tweede byte van de instructie (de vector "v") doet dienst als "masker" en kan bijzonder goed gebruikt worden voor het plaatsen van specifieke bits in de 1-stand.

	Hex. Code	Binaire Code
SET FLAG:	H'7460'	01110110 01000000
SET Inhibit (II):	H'7620'	01110110 00100000

De operatiecode luidt:
H'76'

Binaire Code



Executietijd: 3 cycli (9 klokperioden)

PSW-bits die beïnvloed worden: CC, IDC, RS, WC, OVF, COM, C

Beïnvloeding Conditie-Code: de CC-bits kunnen d.m.v. deze instructie 1 gemaakt worden (SET).

Beschrijving

Deze instructie ter grootte van 2 bytes selecteert m.b.v. enen in de vector "v" de bits van het PSW Lower (PSL) die in de 1-stand gezet moeten worden. Enkele vaak voorkomende combinaties zijn:

	Hex. Code	Binaire Code
Selecteer Register Bank 1:	H′ 7710′	01110111 00010000
Set gebruik van CARRY: WC = 1	H′ 7708′	01110111 00001000
Set logisch vergelijken: COM = 1	H′ 7702′	01110111 00000010
Set CARRY-bit: C = 1	H′ 7701′	01110111 00000001

De operatiecode luidt:
H′ 77′

Binaire Code

Executietijd: 3 cycli (9 klokperioden)

PSW-bits die beïnvloed worden: CC

Beïnvloeding Conditie-Code:	CC1	CC0
alle geselecteerde PSU-bits zijn 1	0	0
niet alle geselect. PSU-bits zijn 1	1	0

Beschrijving

Deze instructie ter grootte van 1 byte selecteert m.b.v. enen welke bit in het PSW Upper (PSU) getoetst dient te worden. De enen zijn weergegeven in de tweede byte, die als "selectie vector" dienst doet. Indien alle geselecteerde bits 1 zijn, dan wordt de CC = 00. Zijn niet alle gekozen bits 1, dan wordt de CC = 10.

Door het toetsen van een patroon bestaat dus vaak onzekerheid over de inhoud van een bepaalde bit. Men kan alleen verifiëren of een bepaald patroon van het PSW overeenkomt met een gewenst patroon. Deze instructie wordt vaak gebruikt om één bepaalde bit van het PSU te toetsen, en niet een combinatie van bits. Enkele veel gebruikte toetsen zijn de volgende:

	Hex. Code	Binaire Code
TEST SENSE	H' B480'	10110100 10000000

Deze opdracht toetst de SENSE-bit, d.w.z. de SENSE-pen die met de buitenwereld verbonden is. Bij een dergelijke verificatie treedt er dus geen menging op van verschillende toetspatronen, daar slechts één PSW-bit onderzocht wordt.

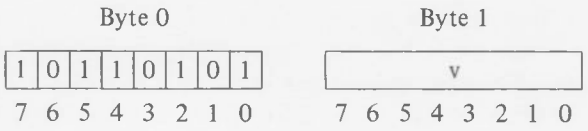
TEST FLAG bit	H' B440'	10110100 01000000

Hierdoor kan het programma toetsen of in een ander deel van het programma, b.v. ten gevolge van een invoer, de FLAG-bit in de 1-stand gezet is. Dit kan noodzakelijk zijn om hem – afhankelijk van de conditie die hij heeft – in de 0-stand te brengen. Staat de bit al in de 0-stand, dan kan het zijn dat hij in de 1-stand gebracht moet worden. Een voorbeeld hiervan is een "impulse-trein", die geschakeld wordt naar aanleiding van een 50 Hz interrupt.

TEST Inhibit (II)	H' B420'	10110100 00100000

De operatiecode luidt:
H' B4'

Binaire Code



Executietijd: 3 cycli (9 klokperioden)

PSW-bits die beïnvloed worden: CC

Beïnvloeding Conditie-Code:	CC1	CC0
alle bits in het geselecteerde PSL zijn 1	0	0
niet alle bits in het geselect. PSL zijn 1	1	0

Beschrijving

Deze instructie ter grootte van 2 bytes toetst de individuele bits van het PSW Lower (PSL), en wel op die plaatsen waar een 1 voorkomt in de vector "v", d.w.z. de tweede byte van de instructie.

Met behulp van deze instructie kunnen bepaalde bits in het PSL getoetst worden. Enkele voorbeelden hiervan zijn:

	Hex. Code	Binaire Code
TEST Inter Digit Carry (IDC)	H' B520'	10110101 00100000
TEST for Overflow (OVF)	H' B504'	10110101 00000100
TEST for Carry (C)	H' B501'	10110101 00000001

De Conditie Code (CC) kan op een speciale manier getoetst worden, namelijk of hij precies overeenkomt met een gegeven waarde. De BRANCH-instructies leveren echter meer mogelijkheden om de CC te verifiëren, en daarom wordt deze wijze van toetsen van de CC-bits hier niet behandeld.

De operatiecode luidt:
H' B5'

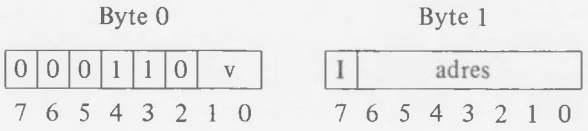
In programma's komen vaak *sprongopdrachten* (*branching, jumping*) voor. De instructieset van de microprocessor 2650 kent een groot aantal sprongopdrachten. De adressering kan direct of indirect zijn, afhankelijk van de Index Control bits (IC) en relatief dan wel absoluut. Veel van de sprongopdrachten toetsen de Conditie-Code, en als die overeenkomt met de voorgeschreven waarde ("true"), kan er een sprong gemaakt worden. De meeste van deze sprongopdrachten hebben een tegenhanger, waarbij de sprong juist gemaakt wordt als de CC niet overeenkomt ("false"). Hierdoor is een groot repertoire van spronginstructies ontstaan.

Elke conditionele spronginstructie kan ook niet-conditioneel gemaakt worden, door 11 als Conditie-Code voor te schrijven. Deze CC bestaat niet, en zal een sprong tot resultaat hebben.

Naast de normale sprongopdrachten zijn er ook een aantal die tevens boek houden van het aantal malen dat de conditionele sprong wordt uitgevoerd. De boekhouding kan bestaan uit het verhogen van een register, resp. uit het verlagen ervan. Bij deze instructies wordt niet getoetst op de Conditie-Code, maar uitsluitend op de inhoud van het register waarin de boekhouding wordt bijgehouden. Deze instructies zijn de instructies BRANCH ON INCREMENT REGISTER en BRANCH ON DECREMENT REGISTER.

Voor operaties waarbij meer dan 1 byte betrokken zijn, zal de boekhouding anders verlopen dan het incrementeren of decrementeren m.b.v. 1 byte. Voor deze gevallen is er ook een sprongopdracht op de inhoud van een register. Dit register doet dan dienst voor de boekhouding. Voor gecompliceerde sprongopdrachten kan gebruik gemaakt worden van een spronginstructie, waarbij het sprongadres in een tabel is ondergebracht. Hierdoor wordt de mogelijkheid geboden tot een meervoudige BRANCH. Een speciale spronginstructie is die waarbij het sprongadres of het indirecte sprongadres zich bevindt op pagina 0, en wel in de eerste 64 bytes van deze pagina of de laatste 64 bytes van deze pagina.

Binaire Code



Executietijd: 3 cycli (9 klokperioden) bij directe adressering
5 cycli (15 klokperioden) bij indirecte adressering

PSW-bits die beïnvloed worden: geen

Directe adressering:

I=0
CC=v
IAR ← IAR+2
IAR ← IAR+adres

CC≠v
IAR ← IAR+2

Indirecte adressering:

I=1
CC=v
IAR ← IAR+2
IAR ← MEM[IAR+adres]

CC≠v
IAR ← IAR+2

Beschrijving

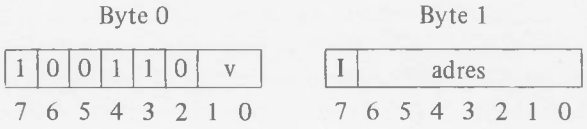
Deze conditionele BRANCH-instructie ter grootte van 2 bytes zorgt ervoor dat het veld "v" wordt vergeleken met de Conditie-Code. Komen deze overeen, dan wordt gesprongen naar het relatieve adres, aangegeven door het Instructie-Adres-Register (IAR) en het veld "a". Dit is het effectieve adres; "a" is in het 2-complement binaire talstelsel. Ingeval er indirecte adressering is, treft men hier de eerste van de 2 bytes aan van het effectieve adres. Als het veld "v" twee enen bevat, wordt er een niet-conditionele sprong uitgevoerd.

Deze instructie is bijzonder geschikt voor sprongopdrachten die voorkomen in een stroomdiagram zoals in fig. 15 is voorgesteld.

De operatiecodes luiden:

Conditie Code	00	H'18'
" "	01	H'19'
" "	10	H'1A'
" "	11	H'1B'

Binaire Code



Executietijd: 3 cycli (9 klokperioden) bij directe adressering
5 cycli (15 klokperioden) bij indirecte adressering

PSW-bits die beïnvloed worden: geen

Directe adressering:

I=0
CC≠v
IAR ← IAR+2
IAR ← IAR+adres

CC=v
IAR ← IAR+2

Indirecte adressering:

I=1
CC≠v
IAR ← IAR+2
IAR ← MEM[IAR+adres]

CC=v
IAR ← IAR+2

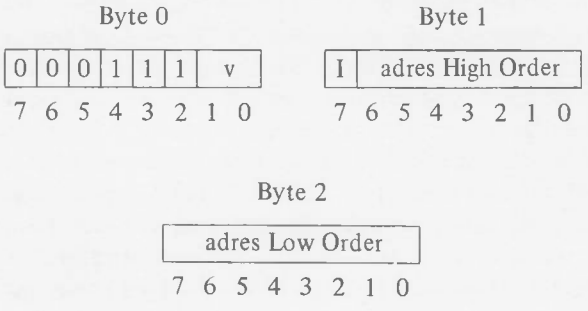
Beschrijving
Deze conditionele BRANCH-instructie ter grootte van 2 bytes zorgt ervoor dat het veld "v" wordt vergeleken met de Conditie-Code. Komen deze beide niet overeen, dan wordt een sprong gemaakt naar het effectieve adres, aangegeven door het veld "a" en het Instructie-Adres-Register (IAR); "a" is in het 2-complement binaire talstelsel. Het veld "v" mag nimmer 11 zijn, omdat het dan als een andere instructie wordt opgevat. Bij indirecte adressering wordt de eerste byte van het effectieve adres aangewezen. De instructie is bijzonder geschikt voor toepassing in een van de stroomdiagrammen (zie fig. 16).

Opmerking:
Deze instructie is complementair in de beslissing met de voorgaande instructie, waarin "true" wordt gebruikt. Dit paar instructies is bijzonder waardevol, omdat men aldus, bij een gegeven Conditie-Code, al dan niet springen kan.

De operatiecodes luiden:

Conditie Code	00	H'98'
" "	01	H'99'
" "	10	H'9A'
" "	11	—

Binaire Code



Executietijd: 3 cycli (9 klokperioden) bij directe adressering
5 cycli (15 klokperioden) bij indirecte adressering

PSW-bits die beïnvloed worden: geen

Directe adressering:

I=0
CC=v
IAR ← IAR+3
IAR ← adres

CC≠v
IAR ← IAR+3

Indirecte adressering:

I=1
CC=v
IAR ← IAR+3
IAR ← MEM[a]

CC≠v
IAR ← IAR+3

Beschrijving

Deze conditionele BRANCH-instructie ter grootte van 2 bytes vergelijkt de inhoud van het veld "v" met de Conditie-Code. Zijn beide identiek, dan wordt een sprong gemaakt naar het veld, aangegeven door de 15 bits van byte 1 en byte 2. Was het bij de voorgaande twee relatieve instructies noodzakelijk binnen 64 bytes in het programma terug te keren, bij deze methodiek kan men elke sprong in het programma maken. De 15 bits staan er borg voor dat elke plaats in het geheugen bereikbaar is.

Wenst men de plaats waar men het programma vervolgt, afhankelijk te maken van de resultaten in een berekening, dan kan de indirecte adressering hier een mogelijkheid bieden. De byte 1 en byte 2 verwijzen naar het indirecte adres. Dit komt overeen met het stroomdiagram van fig. 15, waarbij echter de beperking van de 64 en 62 bytes niet geldt.

Met behulp van indirecte adressering kan men desgewenst een "switch" uitvoeren. Tijdens het programma kan men de plaats, aangegeven door byte 1 en byte 2 (die zich dan in het RAM bevinden), laden met een adres. Men heeft dus een zogenaamde dynamische sprongopdracht. D.m.v. de indirecte adressering kan men nu naar dit adres springen om daar het effectieve adres, d.w.z het uiteindelijke doel waar het programma vervolgd moet worden, aan te treffen.

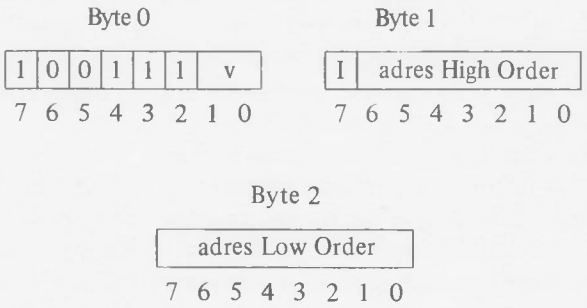
Opmerking:

Indien in het veld "v" 11 wordt aangebracht, dan is dit een niet-conditionele sprongopdracht.

De operatiecodes luiden:

Conditie Code	00	H'1C'
" "	01	H'1D'
" "	10	H'1E'
" "	11	H'1F'

Binaire Code



Executietijd: 3 cycli (9 klokperioden) bij directe adressering
5 cycli (15 klokperioden) bij indirecte adressering

PSW-bits die beïnvloed worden: geen

Directe adressering:

I=0
CC≠v
IAR ← IAR+3
IAR ← adres

CC=v
IAR ← IAR+3

Indirecte adressering:

I=1
CC≠v
IAR ← IAR+3
IAR ← MEM[adres]

CC=v
IAR ← IAR+3

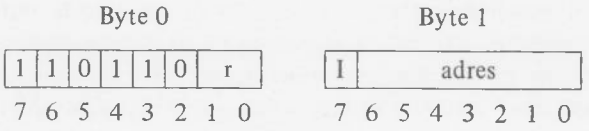
Beschrijving

Deze conditionele BRANCH-instructie ter grootte van 3 bytes vergelijkt het veld "v" met de Conditie-Code. Zijn deze ongelijk dan wordt gesprongen naar het aangegeven adres (de 15 bits van byte 1 en byte 2). Deze sprongopdracht leent zich bij uitstek voor sprongen, zoals aangegeven in fig. 16, maar de beperking van het aantal bytes geldt hier niet. De sprongopdracht is complementair aan de voorgaande absolute sprongopdracht, waarbij de conditie "true" werd gebruikt. Ook bij deze sprongopdracht is het mogelijk een "switch" te maken, door tijdens het programmeren in het adres, aangegeven in byte 1 en byte 2, in het RAM een adres aan te brengen. Bij indirecte adressering kan men op dit adres de eerste byte van het effectieve adres aantreffen. Door op dit adres (als het in RAM is) tijdens het programma een waarde te plaatsen, kan men een dynamische sprongopdracht realizeren (zie vorige instructie).

De operatiecodes luiden:

Conditie Code	00	H'9C'
" "	01	H'9D'
" "	10	H'9E'
" "	11	—

Binaire Code



Executietijd: 3 cycli (9 klokperioden) bij directe adressering
5 cycli (15 klokperioden) bij indirecte adressering

PSW-bits die beïnvloed worden: geen

Directe adressering:

I=0
IAR ← IAR+2
r ← r+1

als R≠0 dan:
IAR ← IAR+adres

als r=0 dan:
IAR ongewijzigd

Indirecte adressering:

I=1
IAR ← IAR+2
r ← r+1

als R≠0 dan:
IAR ← MEM[IAR+adres]

als r=0 dan:
IAR ongewijzigd

Beschrijving

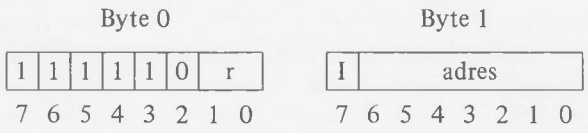
Deze BRANCH-instructie ter grootte van 2 bytes verhoogt bij de uitvoering ervan allereerst de inhoud van het register r met 1. Is de nieuwe waarde in dit register ongelijk 0, dan wordt een sprong gemaakt, overeenkomstig het adres, zoals dit in byte 2 is aangegeven. Deze sprong is relatief t.o.v. de instructie en maakt dan gebruik van het IAR.

Deze instructie is bijzonder geschikt voor sprongen waarbij de boekhouding in het register r moet worden bijgehouden. Voorbeelden daarvan zijn "lussen", die een bepaald aantal malen moeten worden doorlopen. Een ander voorbeeld is het sommeren van de waarden in een tabel, waarbij dan tevens het register r als indexregister kan worden gebruikt (zie fig. 17).

De operatiecodes luiden:

register R0	H'D8'
register R1	H'D9'
register R2	H'DA'
register R3	H'DB'

Binaire Code



Executietijd: 3 cycli (9 klokperioden) bij directe adressering
5 cycli (15 klokperioden) bij indirecte adressering

PSW-bits die beïnvloed worden: geen

Directe adressering:

I=0
IAR ← IAR+2
r ← r−1

als r=0 dan:
IAR ← IAR+adres

als r≠0 dan:
IAR ongewijzigd

Indirecte adressering:

I=1
IAR ← IAR+2
r ← r−1

als r=0 dan:
IAR ← MEM[IAR+adres]

als r≠0 dan:
IAR ongewijzigd

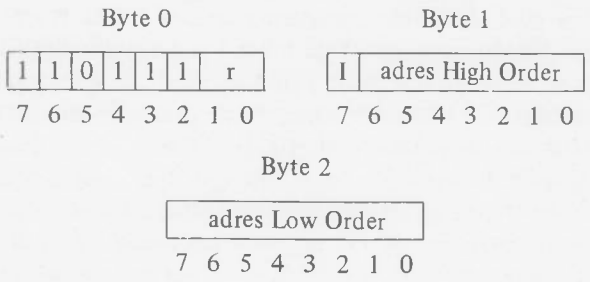
Beschrijving

Deze BRANCH-instructie ter grootte van 2 bytes vermindert de inhoud van het aangegeven register r met 1. Vervolgens wordt nagegaan of de inhoud van register r gelijk aan 0 is. Blijkt dit niet zo te zijn, dan wordt de sprong gemaakt naar het effectieve adres, in dit geval gegeven door de stand van het IAR en de waarde "adres". M.b.v. deze sprongopdracht kan men wederom een "lus" een aantal malen doorlopen en aftellen hoe veel malen dit is geschied. Daartoe plaatst men in het register r het aantal malen dat de lus doorlopen dient te worden. Voor het stroomdiagram van de instructie zie fig. 18.

De operatiecodes luiden:

register R0	H'F8'
register R1	H'F9'
register R2	H'FA'
register R3	H'FB'

Binaire Code



Executietijd: 3 cycli (9 klokperioden) bij directe adressering
5 cycli (15 klokperioden) bij indirecte adressering

PSW-bits die beïnvloed worden: geen

Directe adressering:

I=0
IAR ← IAR+3
r ← r+1

als r≠0 dan:
IAR ← adres

als r=0 dan:
IAR ongewijzigd

Indirecte adressering:

I=1
IAR ← IAR+3
r ← r+1

als r≠0 dan:
IAR ← MEM[adres]

als r=0 dan:
IAR ongewijzigd

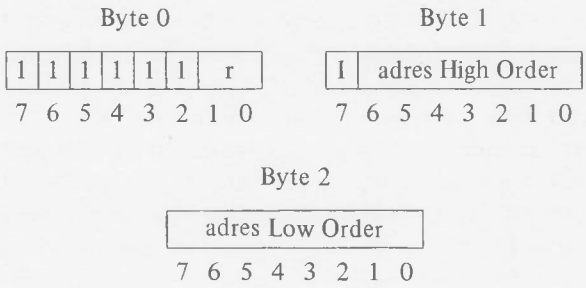
Beschrijving

Deze BRANCH-instructie ter grootte van 3 bytes vermeerdert de inhoud van het register r met 1 en toetst vervolgens de inhoud. Is de inhoud ongelijk aan 0, dan wordt een sprong gemaakt naar het adres, zoals aangegeven in byte 1 en byte 2. Bij indirecte adressering wordt op deze plaats de eerste van de twee bytes aangetroffen van het effectieve adres. Het gebruik van deze instructie komt overeen met die van fig. 17, maar nu kunnen de blokken in wezen elke willekeurige lengte bezitten.

De operatiecodes luiden:

register R0	H'DC'
register R1	H'DD'
register R2	H'DE'
register R3	H'DF'

Binaire Code



Executietijd: 3 cycli (9 klokperioden) bij directe adressering
5 cycli (15 klokperioden) bij indirecte adressering

PSW-bits die beïnvloed worden: geen

Directe adressering:

I=0
IAR ← IAR+3
r ← r−1

als r≠0 dan:
IAR ← adres

als r=0 dan:
IAR ongewijzigd

Indirecte adressering:

I=1
IAR ← IAR+3
r ← r−1

als r≠0 dan:
IAR ← MEM[adres]

als r=0 dan:
IAR ongewijzigd

Beschrijving

Deze BRANCH-instructie ter grootte van 3 bytes vermindert de inhoud van het register r met 1 en toetst vervolgens de inhoud. Is de inhoud van het register r dan gelijk aan 0, dan wordt het programma vervolgd. Is de inhoud evenwel ongelijk aan 0, dan wordt gesprongen naar het adres, aangegeven in de 15 bits van byte 1 en byte 2 (zie fig. 18, maar nu zonder de beperking van de bloklengte).

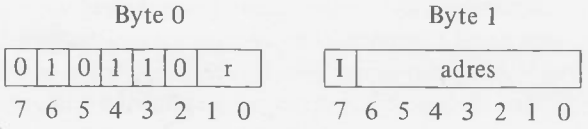
De instructie leent zich voor indirecte adressering en kan als zodanig ook voor dynamische adressering gebruikt worden als men het indirecte adres in het RAM heeft ondergebracht.

Verder leent de instructie zich zeer goed voor het verwerken van tabellen, waarbij register r dan tevens als indexregister gebruikt kan worden.

De operatiecodes luiden:

register R0	H'FC'
register R1	H'FD'
register R2	H'FE'
register R3	H'FF'

Binaire Code



Executietijd: 3 cycli (9 klokperioden) bij directe adressering
5 cycli (15 klokperioden) bij indirecte adressering

PSW-bits die beïnvloed worden: geen

Directe adressering:

I=0
IAR ← IAR+2

als r≠0 dan:
IAR ← IAR+adres

als r=0 dan:
IAR ongewijzigd

Indirecte adressering:

I=1
IAR ← IAR+2

als r≠0 dan:
IAR ← MEM[IAR+adres]

als r=0 dan:
IAR ongewijzigd

Beschrijving

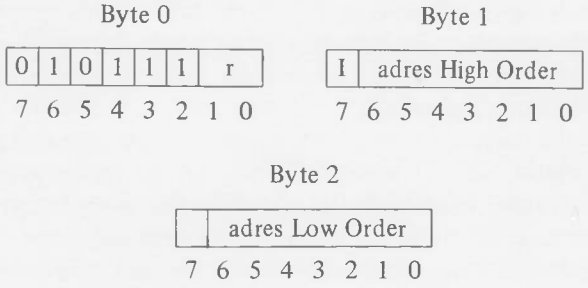
Deze BRANCH-instructie ter grootte van 2 bytes onderzoekt de inhoud van het register r. Indien deze inhoud niet de waarde 0 heeft, dan wordt een sprong gemaakt overeenkomstig de aanwijzingen van het adres in byte 1. Deze instructie heeft t.o.v. de voorgaande instructies het voordeel dat in het programma een ander increment of decrement kan worden aangebracht dan de waarde 1. Men moet er wel voor zorgen dat, ter beeindiging van een lus, de waarde van het register r "0" dient te zijn. Begint men dus met een oneven waarde in het register en men trekt daarvan b.v. het getal 2 af, dan zal men nimmer uit de lus kunnen ontsnappen. Als het register r de waarde 0 heeft, wordt de volgende instructie uitgevoerd.

Deze instructie biedt de mogelijkheid voor indirecte adressering. Op het adres, gegeven door "adres", treft men dan het effectieve adres aan.

De operatiecodes luiden:

register R0	H'58'
register R1	H'59'
register R2	H'5A'
register R3	H'5B'

Binaire Code



Executietijd: 3 cycli (9 klokperioden) bij directe adressering
5 cycli (15 klokperioden) bij indirecte adressering

PSW-bits die beïnvloed worden: geen

Directe adressering:

I=0
IAR ← IAR+3

als r≠0 dan:
IAR ← adres

als r=0 dan:
IAR ongewijzigd

Indirecte adressering:

I=1
IAR ← IAR+3

als r≠0 dan:
IAR ← MEM[adres]

als r=0 dan:
IAR ongewijzigd

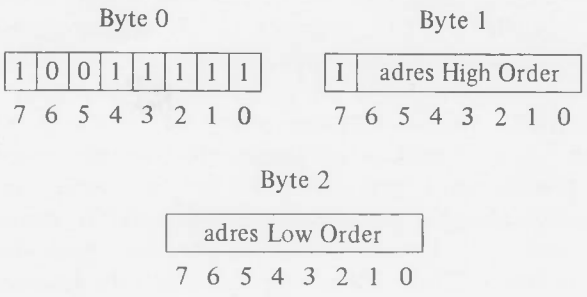
Beschrijving

Deze BRANCH-instructie ter grootte van 3 bytes onderzoekt de inhoud van het register r. Is deze inhoud ongelijk aan 0, dan wordt een sprong gemaakt naar het in byte 1 en byte 2 aangegeven adres. Als de inhoud van het register 0 is, wordt de volgende instructie uitgevoerd. Deze instructie is van belang als men andere increment- of decrementwaarden gebruikt dan 1. Men dient er wel voor te zorgen dat het register r tijdens het uitvoeren van een instructie "0" wordt, daar men anders niet meer uit de lus kan ontsnappen. Met behulp van indirecte adressering kan naar elke gewenste plaats gesprongen worden. Is het indirecte adres in het RAM aanwezig, dan is een dynamische adressering mogelijk, waarbij het van het programma kan afhangen welke waarde op deze plaats geregistreerd staat.

De operatiecodes luiden:

register R0	H'5C'
register R1	H'5D'
register R2	H'5E'
register R3	H'5F'

Binaire Code



Executietijd: 3 cycli (9 klokperioden) bij directe adressering
5 cycli (15 klokperioden) bij indirecte adressering

PSW-bits die beïnvloed worden: geen

Directe adressering:

I=0
IAR ← IAR+3
IAR ← R[3]+adres

Indirecte adressering:

I=1
IAR ← IAR+3
IAR ← R[3]+MEM[adres]

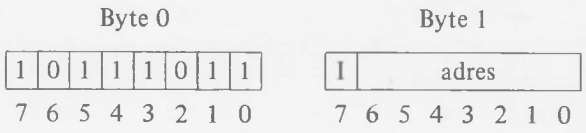
Beschrijving

Deze BRANCH-instructie ter grootte van 3 bytes bewerkstelligt dat in het programma een niet-conditionele sprong wordt gemaakt. Het effectieve adres wordt gegeven door het adres in byte 1 en byte 2 te vermeerderen met de inhoud van het register R3.

Deze instructie is bijzonder waardevol, daar men m.b.v. de inhoud van het register R3 naar diverse programma's kan springen, afhankelijk van de waarde die tijdens het programma in dit register is geladen. Deze methode is nog effectiever als de sprongadressen worden verenigd in een tabel waarvan de basis aangegeven wordt door byte 1 en byte 2. Bij indirecte adressering wordt in dat geval de inhoud van het indexregister bij het adres (was aanwezig in byte 1 en byte 2) opgeteld, en op de nieuwe plaats wordt het effectieve adres gevonden. Ook hierdoor is een "multiple branch" naar elke plaats in het geheugen mogelijk geworden. Bij het toewijzen van de waarden in deze tabel dient men er wel rekening mee te houden dat hier absolute adressen vermeld staan, en dat dus 2 bytes per adres vereist zijn.

De operatiecode luidt:
H' 9F'

Binaire Code



Executietijd: 3 cycli (9klokperioden) bij directe adressering
5 cycli (15 klokperioden) bij indirecte adressering

PSW-bits die beïnvloed worden: geen

Beschrijving

Deze BRANCH-instructie ter grootte van 2 bytes schrijft een niet-conditionele relatieve sprong voor. De oorsprong voor de relatieve verplaatsing is hier echter niet het Instructie-Adres-Register (IAR), maar plaats 0 van het geheugen. Het gebied dat deze instructie bestrijken kan, is dus van 0 t/m +63 en van 8184 t/m 0, d.w.z. dat het adres ook relatief is bij negatieve waarden van "adres" t.o.v. de top van pagina 0. Deze top draagt het adres 8191.

De instructie is vooral van nut als hij gebruikt wordt bij een indirecte adressering. Men kan nu in dit gebied van 128 bytes een aantal adressen aanbrengen en op een betrekkelijk eenvoudige wijze vanuit verschillende punten sprongen veroorzaken naar delen in het geheugen. Elke sprong kost in dit geval slechts 2 bytes. Dit is vooral waardevol als het desbetreffende adres een aantal malen in sprongopdrachten voorkomt.

De operatiecode luidt:

H' 9B'

De instructieverzameling van de microprocessor 2650 biedt de mogelijkheid om *sprongopdrachten naar subroutines* uit te voeren waarbij de stand (inhoud) van het Instructie-Adres-Register (IAR) wordt gered. Bij de tot dusver behandelde sprongopdrachten werd de inhoud van het IAR vervangen door het effectieve adres. Daardoor ging de inhoud van het IAR verloren. De groep instructies die hierna worden behandeld, redden de inhoud van het IAR alvorens dat het effectieve adres in dit register geplaatst wordt.

Laten we het voorbeeld van fig. 19 beschouwen. Het programma bestaat uit de programma-module A, die wordt afgesloten met een SUBROUTINE BRANCH-instructie. Stel dat deze BRANCH-instructie niet wordt uitgevoerd. Dan zal het programma verder gaan met het deel in de module B. Als echter wèl een BRANCH-instructie wordt uitgevoerd, dan wordt verdergegaan met de module C, die elders in het geheugen geplaatst is. Aan het einde van de module C heeft men met behulp van de z.g. RETURN-instructie de mogelijkheid om terug te springen naar het oorspronkelijke programma, en wel naar de eerste instructie van de module B. Voor dat doel wordt de eerder geredde inhoud van het IAR opnieuw in het IAR geplaatst. De uitvoering van de eerste instructie van de module B is daarvan dan een logische consequentie. De RETURN-opdracht moet gezien worden als een niet-conditionele sprongopdracht. Het adres dat bij een dergelijke sprongopdracht behoort, is niet aanwezig in de sprongopdracht zelf, maar in de "red"-plaats.

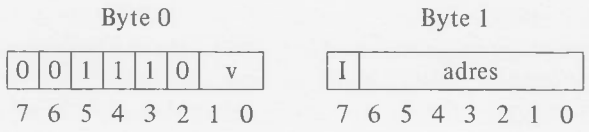
Het voordeel van deze configuratie boven die van normale sprongopdrachten is, dat de module C vele malen gebruikt kan worden. Normaal gesproken zal een programma bij een microprocessor in een ROM aanwezig zijn, hetgeen inhoudt dat men de adressen die bij de spronginstructies staan, niet kan wijzigen. Heeft men de bedoeling om na het doorlopen van een module naar het hoofdprogramma terug te keren, dan dient men dit door het wijzigen van adressen mogelijk te maken. De indirecte adressering zou hiervoor het "gereedschap" kunnen zijn. De methodiek is echter omvangrijk, en het is eenvoudiger hiervoor een speciaal stel instructies te creëren.

De geredde inhoud van het IAR wordt in een speciaal geheugen geplaatst, dat de vorm heeft van een stapelmechanisme. Dit betekent dat de laatst ingebrachte inhoud bovenop de stapel komt te liggen. Wenst men opnieuw terug te keren, dan wordt de inhoud van de top van de stapel gebruikt voor het vullen van het IAR. Vergelijkt men nu deze constructie met b.v. niet-conditionele indirecte spronginstructies, dan ligt hier het indirecte adres vast, namelijk bovenop de stapel (top of stack = TOS), en dit adres blijft de top van de stapel. Waar deze zich precies bevindt, wordt door de stapelteller (stackpointer = SP) in het PSW automatisch bijgehouden; de programmeur behoeft zich hierover geen zorgen te maken. Fig. 20 laat zien hoe in een bepaald programma een andere module C twee malen is betreden en na afloop is deze in beide gevallen teruggekeerd naar de juiste instructie van het hoofdprogramma. Programmamodules zoals C worden vaak subroutines genoemd.

Een subroutine kan op zijn beurt weer een andere subroutine oproepen. In dat geval wordt er namelijk een "subroutine branch" uitgevoerd (al dan niet conditioneel) en de inhoud van het IAR opnieuw gered. Het adres van de volgende instructie in deze subroutine is nu dus in de TOS. De andere subroutine wordt vervolgens betreden en na afloop keert men automatisch terug naar de eerste subroutine (zie fig. 21). Op deze wijze kan men in het geheel 8 subroutines na elkaar aanroepen. Men noemt dit "nesten" van subroutines. In principe is het ook mogelijk de subroutines zelf aan te roepen. Dit is een recursieve techniek.

Bij het betreden van een subroutine dient men te overwegen of het PSW van het oorspronkelijke programma al dan niet gered dient te worden. Dit PSW kan namelijk door de subroutine die men betreden heeft, beïnvloed worden, en het kan wenselijk zijn aan het einde van de subroutine het oorspronkelijke PSW te herstellen. Daartoe dient men in het begin van de subroutine een STORE-opdracht te geven, waardoor de inhoud van het PSW aan het register R0 wordt overgedragen. Hierna kan men met het tweede stel instructies de inhoud van R0 in het geheugen plaatsen. Voordat men terugkeert naar het hoofdprogramma (of naar de subroutine bij meervoudige nesting), dient men ervoor zorg te dragen dat het tegenovergestelde transport geschiedt, d.w.z. van het geheugen naar R0 en vervolgens van R0 naar het PSW. Uitdrukkelijk wordt erop gewezen dat dit in veel gevallen niet noodzakelijk zal zijn. Het is echter wenselijk bij elke sprong naar een subroutine zorgvuldig te overwegen of het PSW al dan niet gered dient te worden.

Binaire Code



Executietijd: 3 cycli (9 klokperioden) bij directe adressering
5 cycli (15 klokperioden) bij indirecte adressering

PSW-bits die beïnvloed worden: SP

Directe adressering:

I=0
CC=v
IAR ← IAR+2
TOS ← IAR
IAR ← IAR+adres

CC≠v
IAR ← IAR+2

Indirecte adressering:

I=1
CC=v
IAR ← IAR+2
TOS ← IAR
IAR ← MEM[IAR+adres]

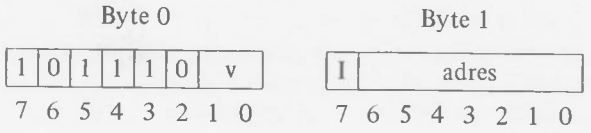
CC≠v
IAR ← IAR+2

Beschrijving
Deze conditionele subroutine branch instructie ter grootte van 2 bytes vergelijkt het veld "v" met de Conditie-Code. Als ze overeenkomen, wordt gesprongen naar het relatieve adres, aangegeven door het Instructie Adres Register (IAR) en het veld "adres". Dit is het effectieve adres. In geval er indirecte adressering is, treft men hier de eerste van de 2 bytes aan van het effectieve adres. Als het veld "v" 2 enen bevat, wordt er een niet-conditionele sprongopdracht uitgevoerd.

De operatiecodes luiden:

Conditie-Code	00	H'38'
"	01	H'39'
"	10	H'3A'
"	11	H'3B'

Binaire Code



Executietijd: 3 cycli (9 klokperioden) bij directe adressering
5 cycli (15 klokperioden) bij indirecte adressering

PSW-bits die beïnvloed worden: SP

Directe adressering:

I=0
CC≠v
IAR ← IAR+2
TOS ← IAR
IAR ← IAR+adres

CC=v
IAR ← IAR+2

Indirecte adressering:

I=1
CC≠v
IAR ← IAR+2
TOS ← IAR
IAR ← MEM[IAR+adres]

CC=v
IAR ← IAR+2

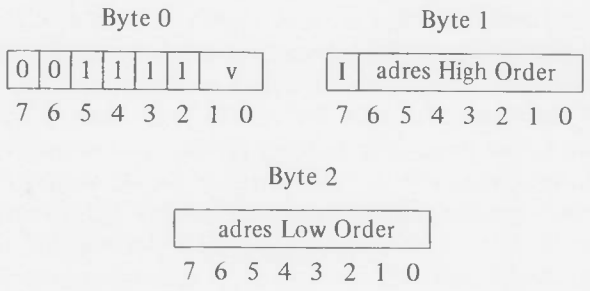
Beschrijving
Deze conditionele branch to subroutine instructie ter grootte van 2 bytes vergelijkt het veld "v" met de Conditie-Code. Komen deze beide niet overeen, dan wordt een sprong gemaakt naar het adres, aangegeven door het veld "adres" en het IAR; "adres" is in het 2-complement binaire talstelsel. Het veld "v" mag nimmer '11' zijn, daar het dan als een andere instructie wordt opgevat. Bij indirecte adressering wordt de eerste byte van het effectieve adres aangewezen.

Opmerking
Deze instructie is complementair in de beslissing met de voorgaande instructie, waarin "true" wordt gebruikt. Dit paar instructies is bijzonder waardevol omdat het de mogelijkheid biedt bij een gegeven Conditie-Code al dan niet te springen.

De operatiecodes luiden:

Conditie Code	00	H'B8'
"	01	H'B9'
"	10	H'BA'
"	11	—

Binaire Code



Executietijd: 3 cycli (9 klokperioden) bij directe adressering
5 cycli (12 klokperioden) bij indirecte adressering

PSW-bits die beïnvloed worden: SP

Directe adressering:

CC=v
IAR ← IAR+3
TOS ← IAR
IAR ← a

CC≠v
IAR ← IAR+3

Indirecte adressering:

CC=v
IAR ← IAR+3
TOS ← IAR
IAR ← MEM[adres]

CC≠v
IAR ← IAR+3

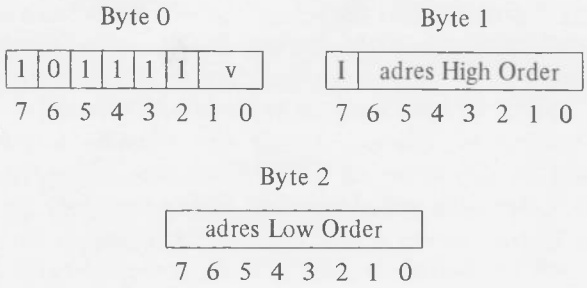
Beschrijving
Deze conditionele branch to subroutine instructie ter grootte van 3 bytes vergelijkt de inhoud van het veld "v" met de Conditie-Code. Zijn beide identiek, dan wordt een sprong gemaakt naar het veld aangegeven door de 15 bits van byte 1 en byte 2. Bij deze methode kan men naar elke plaats in het programma springen. Dit is van bijzonder belang; immers de subroutine zal zelden in dezelfde pagina, laat staan in hetzelfde gebied van 32K bytes beschikbaar zijn, maar meestal in een ander deel van het geheugen. Voor heel veel gevallen is deze absolute adressering belangrijker dan de relatieve adressering, in het bijzonder in die gevallen waarin een "bibliotheek" van subroutines aanwezig is. Als men de plaats waar de subroutine betreden wordt, afhankelijk wenst te maken van het resultaat van een berekening, dan kan in dit geval de indirecte adressering een mogelijkheid bieden. Met behulp van indirecte adressering kan men desgewenst een keuze maken uit verschillende subroutines. Tijdens het programma kan men de plaats (die wordt aangegeven door byte 1 en byte 2 en zich in het RAM bevindt) laden met een adres. Dit adres is dan het adres van de subroutine die men wenst te gebruiken.

Opmerking
Indien in het veld "v" 11 wordt aangebracht, dan is het een niet-conditionele sprongopdracht.

De operatiecodes luiden:

Conditie-Code	00	H'3C'
"	" 01	H'3D'
"	" 10	H'3E'
"	" 11	H'3F'

Binaire Code



Executietijd: 3 cycli (9 klokperioden) bij directe adressering
5 cycli (15 klokperioden) bij indirecte adressering

PSW-bits die beïnvloed worden: SP

Directe adressering:

I=0
CC≠v
IAR ← IAR+3
TOS ← IAR
IAR ← adres

CC=v
IAR ← IAR+3

Indirecte adressering:

I=1
CC≠v
IAR ← IAR+3
TOS ← IAR
IAR ← MEM[adres]

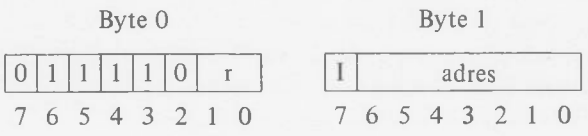
CC=v
IAR ← IAR + 3

Beschrijving
Deze conditionele branch to subroutine instructie ter grootte van 3 bytes vergelijkt het veld "v" met de Conditie-Code. Zijn ze beide niet gelijk, dan wordt gesprongen naar het aangegeven adres (de 15 bits van byte 1 en byte 2). Het voordeel van deze opdracht is dat men de subroutine op elke plaats in het Read Only Memory (ROM) kan aanbrengen. Deze sprongopdracht is complementair aan de voorgaande absolute sprongopdracht, waarbij de conditie "true" werd gebruikt. Ook bij deze sprongopdracht is het mogelijk uit verschillende subroutines een keuze te maken door tijdens het programmeren in het adres, aangegeven in byte 1 en byte 2 van het RAM, een adres van de subroutine aan te brengen.

De operatiecodes luiden:

Conditie-Code	00	H'BC'
"	01	H'BD'
"	10	H'BE'
"	11	—

Binaire Code



Executietijd: 3 cycli (9 klokperioden) bij directe adressering
5 cycli (15 klokperioden) bij indirecte adressering

PSW-bits die beïnvloed worden: SP

Directe adressering:

I=0
IAR ← IAR+2

Als r≠0 dan:
TOS ← IAR
IAR ← IAR+adres

Als r=0 dan:
IAR ongewijzigd

Indirecte adressering:

I=1
IAR ← IAR+2

als r≠0 dan:
TOS ← IAR
IAR ← MEM[IAR+adres]

als r=0 dan:
IAR ongewijzigd

Beschrijving

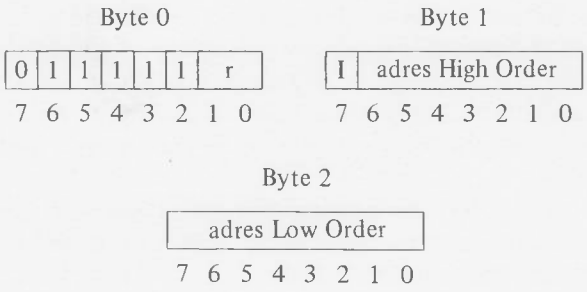
Deze branch to subroutine ter grootte van 2 bytes onderzoekt de inhoud van het register r. Als deze inhoud niet de waarde 0 heeft, dan wordt een sprong gemaakt, overeenkomstig de aanwijzing van het adres in byte 1. Deze instructie maakt het noodzakelijk dat men – alvorens de instructie uit te voeren – het register r op de juiste waarde heeft gebracht.

Door middel van indirecte adressering kan een subroutine worden bereikt die zich ver buiten het gebied van dit programma-onderdeel bevindt.

De operatiecodes luiden:

register	R0	H'78'
register	R1	H'79'
register	R2	H'7A'
register	R3	H'7B'

Binaire Code



Executietijd: 3 cycli (9 klokperioden) bij directe adressering
5 cycli (15 klokperioden) bij indirecte adressering

PSW-bits die beïnvloed worden: SP

Directe adressering:

I=0
IAR ← IAR+3

als r≠0 dan:
TOS ← IAR
IAR ← adres

als r=0 dan:
IAR ongewijzigd

Indirecte adressering:

I=1
IAR ← IAR+3

als r≠0 dan:
TOS ← IAR
IAR ← MEM[adres]

als r=0 dan:
IAR ongewijzigd

Beschrijving

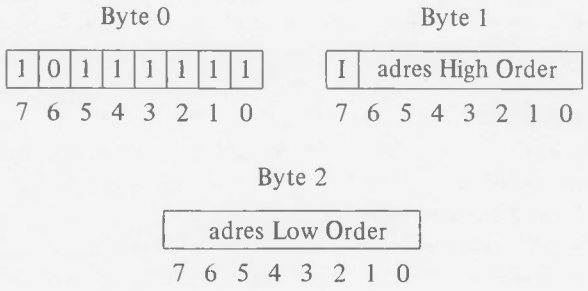
Deze branch to subroutine instructie ter grootte van 3 bytes onderzoekt de inhoud van register r. Is deze inhoud niet 0, dan wordt een sprong gemaakt naar het adres aangegeven in byte 1 en byte 2. Als de inhoud van het register gelijk aan 0 is, dan wordt de volgende instructie uitgevoerd.

Deze instructie is van belang als men een subroutine meer dan één keer wenst te doorlopen. Men dient echter zelf ervoor te zorgen dat de waarde van het register r telkens wordt bijgewerkt. Deze instructie laat zich heel goed combineren met de "branch on register non-zero absolute" (BRNA) of de "branch on register non-zero relative" (BRNR)-instructie. Eerst springt men naar de subroutine, voert deze subroutine uit en keert vervolgens terug, waarna de boekhouding plaatsvindt d.m.v. één van de laatstgenoemde instructies; zie fig. 22

De operatiecodes luiden:

register	R0	H'7C'
register	R1	H'7D'
register	R2	H'7E'
register	R3	H'7F'

Binaire Code



Executietijd: 3 cycli (9 klokperioden) bij directe adressering
5 cycli (15 klokperioden) bij indirecte adressering

PSW-bits die beïnvloed worden: SP

Directe adressering:

I=0
IAR ← IAR+3
TOS ← IAR
IAR ← R[3]+adres

Indirecte adressering:

I=1
IAR ← IAR+3
TOS ← IAR
IAR ← R[3]+MEM[adres]

Beschrijving

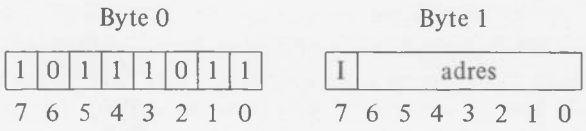
Deze branch to subroutine instructie ter grootte van 3 bytes zorgt ervoor dat in het programma een niet-conditionele sprong wordt gemaakt. Het effectieve adres wordt verkregen door het adres in byte 1 en byte 2 te vermeerderen met de inhoud van register R3. Deze instructie is bijzonder waardevol, daar men m.b.v. de inhoud van register R3 naar diverse subroutines kan springen, afhankelijk van de waarde die tijdens het programma in dit register is geladen. Deze methode is nog effectiever als de sprongadressen in een tabel zijn verenigd waarvan de basis wordt aangegeven door byte 1 en byte 2. Bij indirecte adressering wordt in dat geval de inhoud van het indexregister bij het adres in byte 1 en byte 2 opgeteld; op de nieuwe plaats wordt het effectieve adres dan gevonden.

Opmerking

In deze instructie wordt de verplaatsing dus uitgevoerd voordat het effectieve adres wordt bepaald. Door deze methodiek is een multipele selectie van subroutines mogelijk geworden d.m.v. een tabel. Bij het toewijzen van waarden in deze tabel, moet men er wel rekening mee houden dat er absolute adressen staan, zodat ze 2 bytes per adres vereisen.

De operatiecode luidt:
H′ BF′

Binaire Code



Executietijd: 3 cycli (9 klokperioden) bij directe adressering
5 cycli (15 klokperioden) bij indirecte adressering

PSW-bits die beïnvloed worden: SP

Beschrijving

Deze branch (subroutine) instructie ter grootte van 2 bytes schrijft een niet-conditionele relatieve sprong voor. De oorsprong voor de relatieve verplaatsing is hier echter niet het IAR, maar plaats 0 van het geheugen. Het gebied dat deze instructie bestrijken kan, is dus van 0 t/m +63 en van 8128 t/m 0, d.w.z. dat het adres ook relatief is bij negatieve waarden van "adres" t.o.v. de top van pagina 0. Deze top draagt het adres 8191.

De instructie is vooral dan waardevol als hij gebruikt wordt in combinatie met een indirecte adressering. Men kan dan in het gebied van 128 bytes een aantal adressen aanbrengen en op een betrekkelijk eenvoudige wijze van daaruit sprongen maken naar subroutines elders in het ROM. Elke sprong kost in dit geval slechts 2 bytes. Dit is met name van betekenis als het desbetreffende adres een aantal malen in een sprongopdracht voorkomt. Van bijzonder belang is deze instructie bij de samenwerking met in- en uitvoer-apparatuur. Als een interrupt gehonoreerd wordt (zie de "Hardware Beschrijving"), dan genereert de microprocessor de 2650 zelf de op.code ZBSR. De tweede byte van de op.code moet door de I/O apparatuur via de databus aan de microprocessor worden aangeboden. Daartoe wacht de microprocessor 2650 totdat deze in- en uitvoer-apparatuur de byte aanbiedt. Op deze wijze is het mogelijk om in het gebied van 128 bytes een indirect adres aan te wijzen, en van daar het effectieve adres te verkrijgen waar de (voor de I/O apparatuur) specifieke subroutine aanwezig is.

De operatiecode luidt:
H'BB'

Binaire Code

0	0	0	1	0	1	v	
7	6	5	4	3	2	1	0

Executietijd: 3 cycli (9 klokperioden)

PSW-bits die beïnvloed worden: SP

IAR ← IAR+1

als CC=v:
IAR ← TOS

als CC≠v:
IAR ongewijzigd

Beschrijving
Deze return instructie ter grootte van 1 byte wordt gebruikt om van een subroutine terug te kunnen keren naar het hoofdprogramma. De rest van de subroutine wordt dan verder niet gebruikt. Voor deze sprongopdracht is een adres nodig; dit bevindt zich in het stapelmechanisme, en wel op de top. Als de Conditie-Code niet overeenkomt met de vector "v", dan wordt geen sprong uitgevoerd en de volgende instructie geladen. Om een niet-conditionele sprong uit te kunnen voeren, moet de vector "v" gelijk aan 11 zijn.

De operatiecodes luiden:

Conditie Code	00	H'14'
"	01	H'15'
"	10	H'16'
"	11	H'17'

Binaire Code

0	0	1	1	0	1	v	
7	6	5	4	3	2	1	0

Executietijd: 3 cycli (9 klokperioden)

PSW-bits die beïnvloed worden: SP, II

IAR ← IAR+1

als CC=v:
IAR ← TOS

als CC≠v:
IAR ongewijzigd

Beschrijving

Als bij het interrumperen van het programma door de in- en uitvoer-apparatuur een "zero branch to subroutine" wordt uitgegeven dan gaat dit gepaard met het zetten in de 1-stand van de bit II in het PSW. Zodra de subroutine voor de behandeling van de aanvrage van deze apparatuur gereed is, moet een nieuwe interrupt mogelijk worden gemaakt. Dit zou kunnen gebeuren door, vóór het terugkeren uit het programma, in het PSW deze bit weer in de 0-stand te zetten. Men loopt dan echter het risico dat, onmiddellijk nà deze handeling en vóór de terugkeer, een nieuwe interrupt ontstaat; hierdoor zou het adresstapelmechanisme opnieuw in beslag worden genomen, zonder dat er een terugkeer plaats heeft. Aldus kan het stapelmechanisme zeer snel vol raken. Om dit te voorkomen, is het "resetten" van de bit II en het terugkeren naar het oorspronkelijke programma gecombineerd in één instructie, waardoor de handelingen ondeelbaar zijn geworden. Voor een niet-conditionele terugkeer moet de vector "v" gelijk aan '11' zijn; bij een andere waarde wordt deze vector vergeleken met de Conditie-Code. Indien de Conditie-Code overeenkomt met de vector, wordt naar het oorspronkelijke programma teruggekeerd. Komt de Conditie-Code niet met de vector overeen, dan wordt de volgende instructie uitgevoerd en de bit II niet in de 0-stand gezet.

De operatiecodes luiden:

Conditie Code	00	H'34'
" "	01	H'35'
" "	10	H'36'
" "	11	H'37'

IN- EN UITVOER-INSTRUCTIES

De microprocessor 2650 zal in het algemeen in een omgeving worden opgenomen, waarin hij bestuurlijke taken krijgt te vervullen. De micro-processor kan deze verrichten aan de hand van de informatie die hij uit externe gegevens krijgt; we noemen deze de invoer. De opdrachten tot het verkrijgen van deze invoer worden door de microprocessor zelfstandig gegeven. Voor het verkrijgen van informatie zijn er drie instructies: deze verzekeren dat de informatie van de apparatuur naar de microprocessor 2650 wordt gestuurd. Het communicatiecentrum voor dit informatietransport in deze microprocessor zijn de registers R0 en R1 t/m R3 in bank 0 en bank 1. Er kan ook datatransport van de 2650 naar de in- en uitvoerapparatuur plaats hebben. Voor dat doel zijn er een aantal schrijf(WRITE)-instructies die een byte transporteren van een van de bovengenoemde registers naar het desbetreffende I/O apparaat. Om dit alles te realiseren, beschikken we over drie methoden: de Memory Mapped I/O, de Extended I/O en de Non-Extended I/O.

Memory Mapped I/O

Bij de Memory Mapped I/O wordt de in- en uitvoerapparatuur beschouwd als een onderdeel van het geheugen, d.w.z. dat ze normale geheugenadressen krijgen en dat ze gebruik maken van de lees- en schrijfcommando's, zoals dit ook bij het geheugen geschiedt.

Een nadeel van deze methode is dat een deel van het geheugen, d.w.z. van de 32K bytes, opgeofferd moeten worden voor de communicatie met de I/O apparaten. Dit nadeel geldt uiteraard uitsluitend als het geheugen volledig gebruikt moet worden. Een ander nadeel kan zijn (zie "Hardware Beschrijving") dat de decodering van deze adressen kostbaarder is. Dat geldt echter niet als het geheugen zelf niet meer dan 16K bytes bedraagt. De meest significante adreslijn kan dan gebruikt worden om onderscheid te maken tussen het geheugen en de I/O apparaten.

Het grote voordeel van deze methode is daarin gelegen dat alle instructies waarbij thans de tweede operand in het geheugen aanwezig is, ook van toepassing zijn op de in- en uitvoerapparatuur. Dit betekent b.v. dat men rekenkundig of logische bewerkingen uit kan voeren tussen een operand in de microprocessor 2650 en een operand van het I/O apparaat. Het beschikbare instructierepertoire is dus aanzienlijk groter dan bij de hierna volgende speciale I/O instructies.

Extended I/O

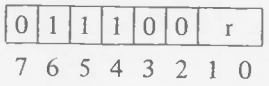
Bij de Extended I/O bestaat een instructie uit twee bytes. De eerste byte bevat de operatiecode en een nummer van een register. Dit register is de plaats van bestemming of de oorsprong van data. De tweede byte van de instructie bevat het adres van het I/O apparaat. Dit houdt in dat er maximaal 256 I/O apparaten geadresseerd kunnen worden. Het adres wordt op de adresbus geplaatst en zal als zodanig dus in het geheugen werkzaam kunnen zijn. De besturingsdraad M/IO instrueert echter de I/O apparaten en het geheugen dat het adres thans bestemd is voor een I/O apparaat en niet voor het geheugen.

Extended I/O instructies zijn uitsluitend bestemd om data naar de microprocessor 2650 te halen (READ) of naar het I/O apparaat te brengen (WRITE). Ze verrichten niet meer dan uitsluitend de communicatie-handelingen. Elke andere actie dient door een van de andere instructies verricht te worden. De instructies zijn echter bijzonder waardevol om b.v. aan I/O apparatuur opdrachten te geven. Ze kunnen dus bij uitstek dienst doen voor besturing en data-overdracht. Men kan er op deze wijze b.v. ook voor zorgen dat I/O apparaten na een zekere tijd een INTERRUPT plegen, waardoor gebruik gemaakt kan worden van de EXTENDED I/O en van de voor hen bestemde subroutines.

Non Extended I/O

Voor de communicatie met de I/O apparatuur zijn nog vier andere instructies aanwezig. Twee ervan hebben betrekking op het zenden van data naar de I/O apparaten en twee ervan op het ontvangen van data. *Er wordt bij deze instructies geen adres verstrekt.* Men dient er dus voor te zorgen dat het desbetreffende apparaat geactiveerd is. Dit kan geschieden door middel van een EXTENDED I/O instructie of doordat een apparaat via een interrupt-mechanisme gekoppeld is met de microprocessor. Een andere mogelijkheid is dat er slechts twee speciale I/O-apparaten zijn, die uitsluitend via de NON-EXTENDED I/O te bereiken zijn. Ze worden dan geselecteerd d.m.v. de z.g. CONTROL- en DATA-draad. Om dit te bereiken, zijn de 4 instructies weer in 2 hoofdgroepen gesplitst. Voor verdere details wordt men verwezen naar: "Hardware van de microprocessor 2650"

Binaire Code



Executietijd: 2 cycli (6 klokperioden)

PSW-bits die beïnvloed worden: CC

Beïnvloeding Conditie-Code:

register r	CC1	CC0
positief	0	1
nul	0	0
negatief	1	0

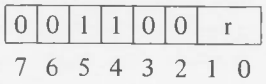
Beschrijving

Deze input-instructie ter grootte van 1 byte heeft tot gevolg dat er communicatie plaats heeft tussen een van te voren aangewezen I/O apparaat en een register in de microprocessor 2650. Dit register is gespecificeerd in het veld "r" van de instructie (bit 0 en bit 1) en afhankelijk van de gekozen registerbank. Voor het uitvoeren van de instructie, plaatst men de OPREQ-lijn samen met de M/$\overline{IO}$-lijn, en de R/$\overline{W}$-lijn in de 1-stand. Gedurende het OPREQ-signaal wordt de D/$\overline{C}$-lijn omgeschakeld naar D (data) en de E/$\overline{NE}$-lijn naar $\overline{NE}$ (non-extended). Dit heeft tot gevolg dat hetzij een bepaald apparaat (gespecificeerd door de D/$\overline{C}$-lijn en de E/$\overline{NE}$-lijn) dan wel een eerder aangewezen I/O apparaat zijn data overdraagt naar de microprocessor 2650.

De operatiecodes luiden:

register R0 H'70'
register R1 H'71'
register R2 H'72'
register R3 H'73'

Binaire Code



Executietijd: 2 cycli (6 klokperioden)

PSW-bits die beïnvloed worden: CC

Beïnvloeding Conditie-Code:

register r	CC1	CC0
positief	0	1
nul	0	0
negatief	1	0

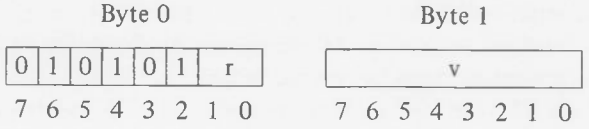
Beschrijving

Deze input-instructie ter grootte van 1 byte draagt een byte over naar een register in de microprocessor 2650. De werking is analoog aan die van de vorige instructie, behalve dat de D/$\overline{C}$-lijn wordt omgeschakeld naar $\overline{C}$ (control). Men kan deze omschakeling gebruiken om een I/O apparaat aan te wijzen. Ook is het mogelijk naar een eerder aangegeven apparaat stuursignalen te zenden.

De operatiecodes luiden:

register	R0	H'30'
register	R1	H'31'
register	R2	H'32'
register	R3	H'33'

Binaire Code



Executietijd: 3 cycli (9 klokperioden)

PSW-bits die beïnvloed worden: CC

Beïnvloeding Conditie-Code:

register r	CC1	CC0
positief	0	1
nul	0	0
negatief	1	0

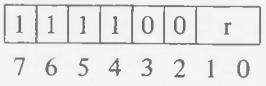
Beschrijving

Deze input-instructie ter grootte van 2 bytes draagt een byte via de databus van een I/O apparaat over naar een register r van de microprocessor 2650. Gedurende deze instructie wordt de tweede byte "v" op de minst significante lijnen van de adresbus geplaatst en door een I/O apparaat geïnterpreteerd als een adres. Dit I/O apparaat zal dan de data op de databus plaatsen. Met behulp van deze instructie kan dus uit een geadresseerd I/O apparaat één byte worden verkregen.

De operatiecodes luiden:

register R0 H'54'
register R1 H'55'
register R2 H'56'
register R3 H'57'

Binaire Code



Executietijd: 2 cycli (6 klokperioden)

PSW-bits die beïnvloed worden: geen

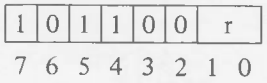
Beschrijving

Deze output-instructie ter grootte van 1 byte draagt een byte uit een register r over naar een I/O apparaat. De selectie van het I/O apparaat dient of tevoren te geschieden d.m.v. een adressering, b.v. van een WRITE EXTENDED instructie, of kan geschieden door een combinatie van de M/$\overline{IO}$, Read/Write, E/$\overline{NE}$ en Data/Control lijnen.

De operatiecodes luiden:

register	R0	H'F0'
register	R1	H'F1'
register	R2	H'F2'
register	R3	H'F3'

Binaire Code



Executietijd: 2 cycli (6 klokperioden)

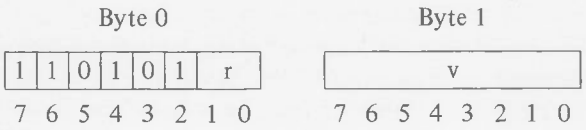
Beschrijving

Deze output-instructie ter grootte van 1 byte draagt een byte van een register r over naar een I/O apparaat. De adressering geschiedt hetzij door van tevoren het apparaat te selecteren, b.v. door een WRITE EXTENDED instructie of door een combinatie van de E/$\overline{NE}$ en de D/$\overline{C}$ lijnen (de laatstgenoemde lijn wordt in de stand $\overline{C}$ (control) gebracht). In dit geval kunnen vier I/O apparaten geadresseerd worden zonder dat een specifiek adres wordt verstrekt.

De operatiecodes luiden:

register	R0	H'B0'
register	R1	H'B1'
register	R2	H'B2'
register	R3	H'B3'

Binaire Code



Executietijd: 3 cycli (9 klokperioden)

PSW-bits die beïnvloed worden: geen

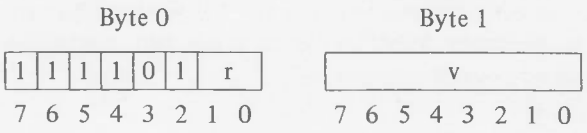
Beschrijving

Deze output-instructie ter grootte van 2 bytes draagt een byte uit een register r over aan een I/O apparaat. De adressering geschiedt door "v" op de minst significante lijnen van de adresbus te plaatsen.

De operatiecodes luiden:

register	R0	H'D4'
register	R1	H'D5'
register	R2	H'D6'
register	R3	H'D7'

Binaire Code



Executietijd: 3 cycli (9 klokperioden)

PSW bits beïnvloeding: CC

Conditie Code beïnvloeding:	CC1	CC0
alle geselecteerde bits zijn "1"	0	0
niet alle geselect. bits zijn "1"	1	0

Beschrijving

Deze 2 bytes instructie test de bits in het aangegeven register r. Welke bits er getest worden, hangt af van de vector "v". Indien op een bepaalde plaats in "v" een 1 voorkomt, dan zal de daarmee corresponderende bit in het register r onderzocht worden. Afhankelijk van de waarde van deze bit (0 of 1) zal de Conditie Code worden gezet. De CC zal 00 zijn indien alle geselecteerde bits 1 zijn. Zo niet, dan zal de CC 10 bedragen.

De operatiecodes luiden:

register	R0	H'F4'
register	R1	H'F5'
register	R2	H'F6'
register	R3	H'F7'

Binaire Code

1	0	0	1	0	1	r	
7	6	5	4	3	2	1	0

Executietijd: 3 cycli (9 klokperioden)

PSW-bits die beïnvloed worden: CC

De Conditie-Code krijgt een "betekenisloze" waarde.

Beschrijving

De microprocessor 2650 kent geen decimale rekenwijze. De optelling en de aftrekking zijn in binaire code. Wordt echter de data in decimale code weergegeven (d.w.z. BCD code, waarbij één byte 2 cijfers van elk 4 bits kan bevatten), dan kan het bij een optelling voorkomen dat er geen "carry's" aanwezig zijn. Na de eerste 4 bits dient immers een "carry" op te treden als de som groter dan of gelijk aan 10 (tien) is; hetzelfde geldt voor de tweede groep van 4 bits. De "carry" treedt pas op als de som groter is dan 15 (binair 1111). Om aan dit bezwaar te ontkomen, is een DECIMAL ADJUST REGISTER instructie aanwezig. Bij een optelling wordt om de "carry" te verkrijgen, een ADD IMMEDIATE (ADDI) met H'66' uitgevoerd. Hierdoor ontstaan de "carry's". Nu is echter het resultaat te groot. Ontstaat een "carry", dan zijn de "carry"-flip-flop en de Inter Digit Carry (IDC)-flip-flop beïnvloed. Als deze in de "1"-stand staan, dan is er 16 overgedragen; 10 hiervan is wenselijk en de 6 is er ingebracht door de ADDI optelling (zie fig. 23). Er is dan geen correctie nodig. Als er echter géén "carry" is ontstaan, dan is van een decimale digit het resultaat 6 te hoog. Men dient nu selectief een 6 af te trekken, echter alléén indien de desbetreffende carry-bit in de 0-stand staat. Dit selectief aftrekken kan geschieden door de DECIMAL ADJUST REGISTER (DAR) instructie. Deze instructie onderzoekt de "carry's" en trekt selectief van de bijbehorende voorgaande 4 bits H'06' af. Bij het aftrekken is een overeenkomstige handeling noodzakelijk. Nu echter dient men eerst de H'66' op te tellen, doch wordt de aftrekking rechtstreeks uitgevoerd, waarna een DAR operatie moet worden uitgevoerd (zie fig. 24).

De operatiecodes luiden:

register	R0	H'94'
register	R1	H'95'
register	R2	H'96'
register	R3	H'97'

REGISTER - ADRESSERING

OP. CODE	00	REG 0 • REG 0
OP. CODE	01	REG 0 • REG 1
OP. CODE	10	REG 0 • REG 2
OP. CODE	11	REG 0 • REG 3

• = EEN TOEGESTANE OPERATIE

Fig. 1

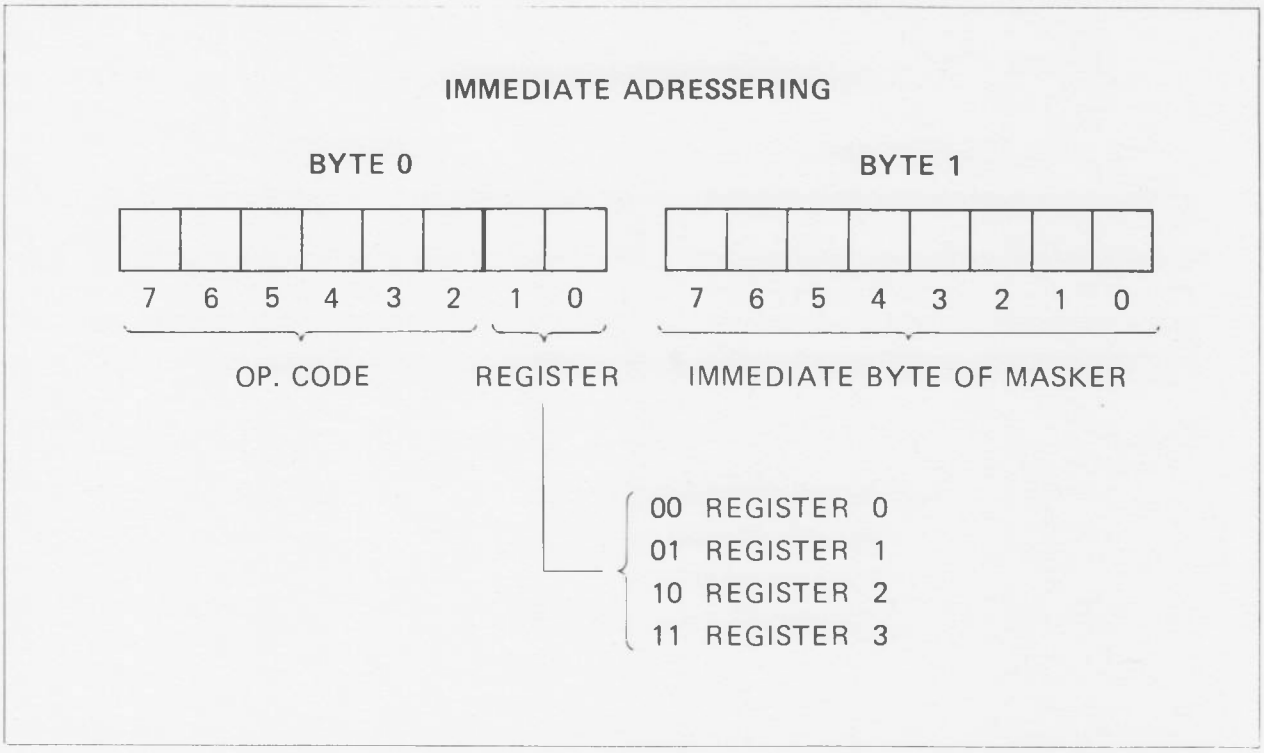

IMMEDIATE ADRESSERING
BYTE 0
BYTE 1
7 6 5 4 3 2 1 0
7 6 5 4 3 2 1 0
OP. CODE
REGISTER
IMMEDIATE BYTE OF MASKER
00 REGISTER 0
01 REGISTER 1
10 REGISTER 2
11 REGISTER 3

Fig. 2

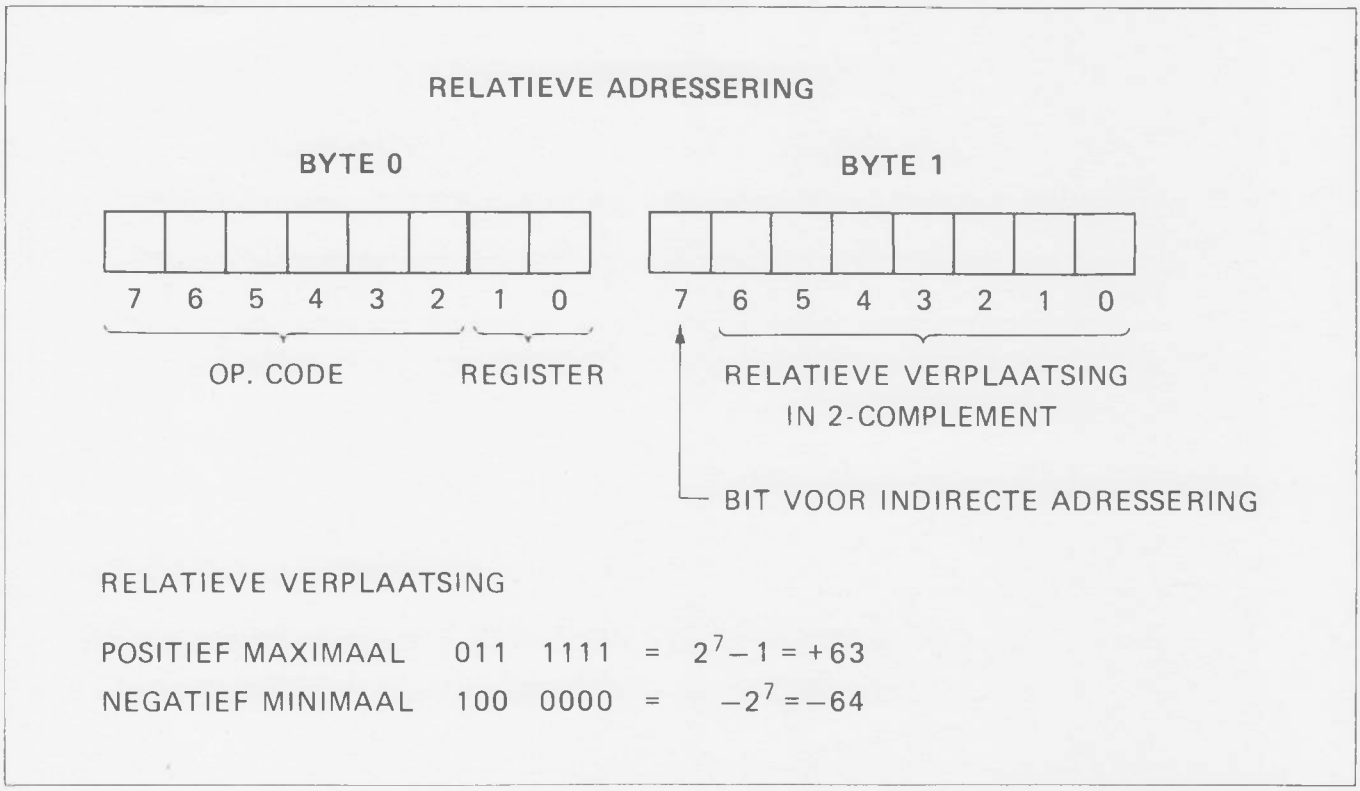

RELATIEVE ADRESSERING
BYTE 0
BYTE 1
7 6 5 4 3 2 1 0
7 6 5 4 3 2 1 0
OP. CODE
REGISTER
RELATIEVE VERPLAATSING
IN 2-COMPLEMENT
BIT VOOR INDIRECTE ADRESSERING
RELATIEVE VERPLAATSING
POSITIEF MAXIMAAL 011 1111 = $2^7-1=+63$
NEGATIEF MINIMAAL 100 0000 = $-2^7=-64$

Fig. 3

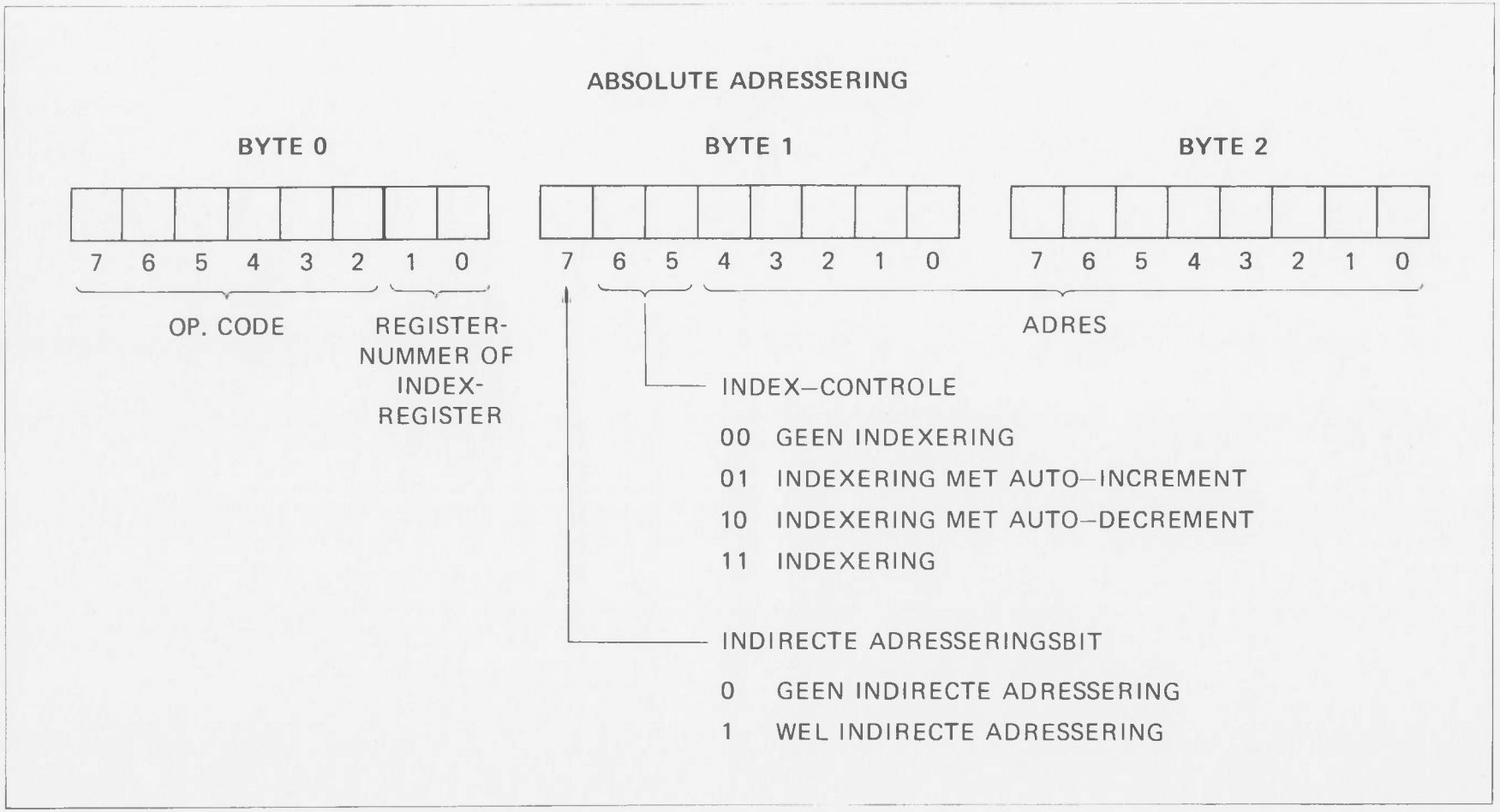
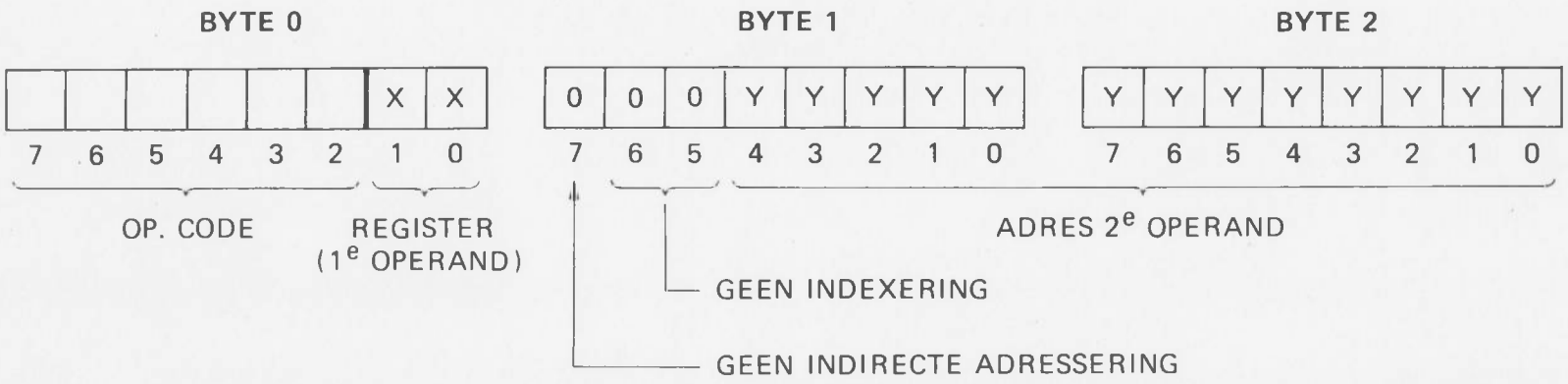

ABSOLUTE ADRESSERING
BYTE 0
BYTE 1
BYTE 2
7 6 5 4 3 2 1 0
7 6 5 4 3 2 1 0
7 6 5 4 3 2 1 0
OP. CODE
REGISTER-NUMMER OF INDEX-REGISTER
ADRES
INDEX–CONTROLE
00 GEEN INDEXERING
01 INDEXERING MET AUTO–INCREMENT
10 INDEXERING MET AUTO–DECREMENT
11 INDEXERING
INDIRECTE ADRESSERINGSBIT
0 GEEN INDIRECTE ADRESSERING
1 WEL INDIRECTE ADRESSERING

Fig. 4

ADRESSERING BIJ NIET–SPRONGINSTRUCTIES



DE 1^{e} OPERAND IS IN REGISTER X X

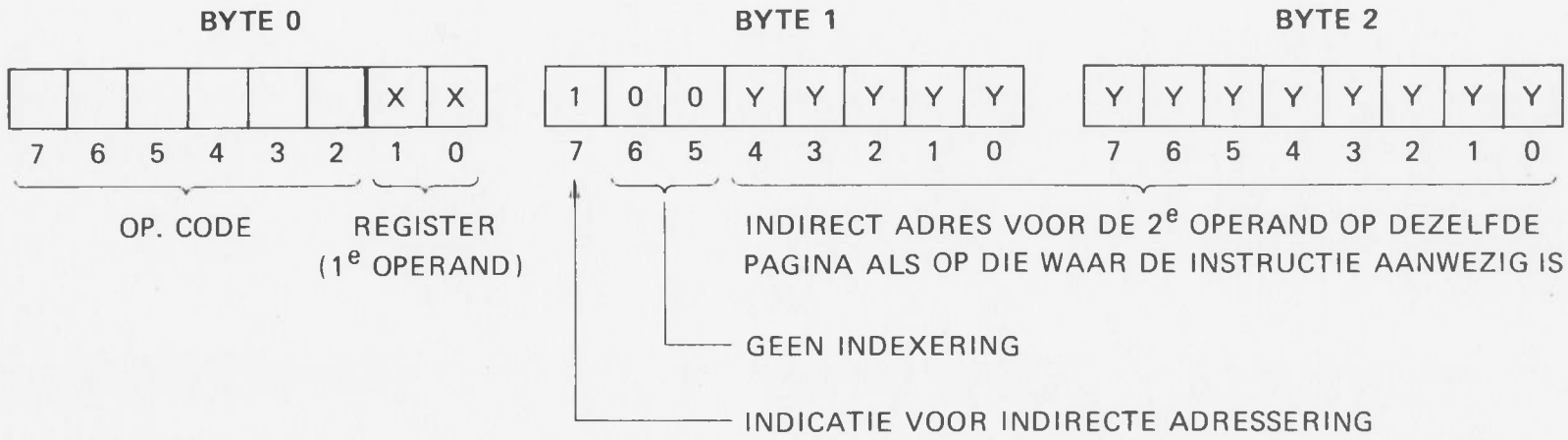
DE 2^{e} OPERAND IS AANWEZIG OP EEN GEHEUGEN PLAATS Y . . . Y

DE ADRESSERING VAN DE 2^{e} OPERAND GESCHIEDT OP DEZELFDE PAGINA ALS OP DIE WAAR DE INSTRUCTIE ZICH BEVINDT

R [X X] ← R [X X] • MEM [PAG, Y . . . Y]

Fig. 5

ABSOLUTE ADRESSERING BIJ NIET–SPRONGINSTRUCTIES



ADRES

[PAG, Y . . . Y] + 1 → MINST SIGNIF. DEEL VAN HET ADRES

[PAG, Y . . . Y] → MEEST SIGNIF. DEEL VAN HET ADRES

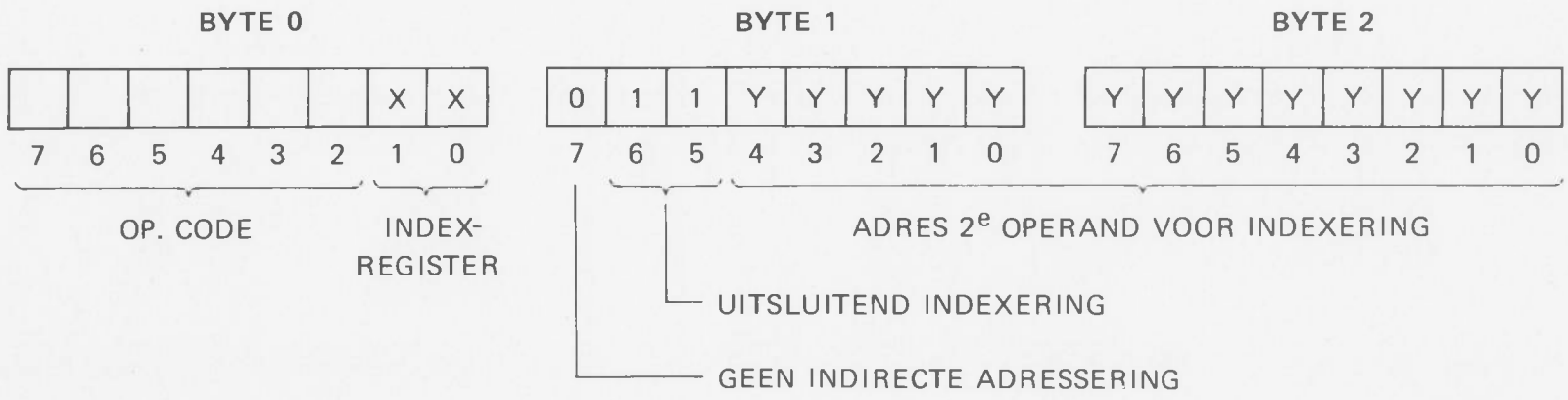
DE 1e OPERAND IS IN REGISTER X X

DE 2e OPERAND WORDT GEGEVEN DOOR HET ADRES OP DE PLAATSEN
[PAG, Y . . . Y] EN [PAG, Y . . . Y] + 1

R [X X] ← R [X X] • MEM [MEM [PAG, Y . . . Y]]

Fig. 6

ABSOLUTE ADRESSERING BIJ NIET–SPRONGINSTRUCTIES



ADRES ← PAG, R [X X] + Y . . . Y

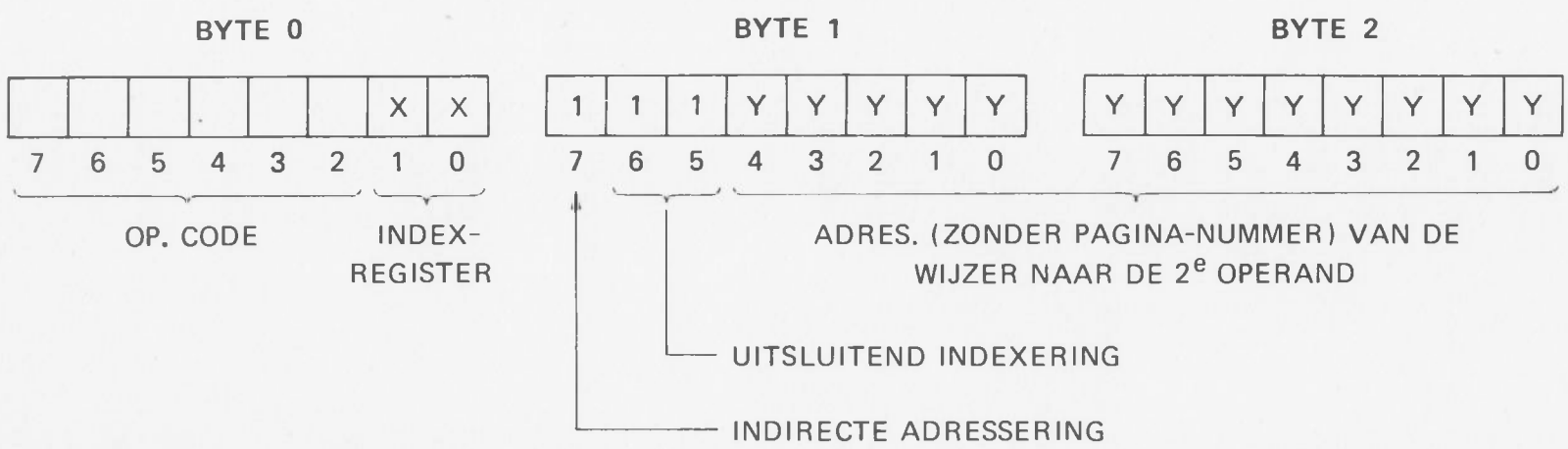
R [0] ← R [0] + MEM [ADRES]

VOORBEELD:

STEL	PAG	=	0 1	= 8.192
STEL	Y . . . Y	=	0 0 0 0 0 0 1 0 0 1 0 1 1	= 75
	R [X X]	=	1 0 1 0 1 0 1 0	= 170
	ADRES 2e OPERAND	=	0 1 0 0 0 0 0 1 1 1 1 0 1 0 1	= 8.437

Fig. 7

ABSOLUTE ADRESSERING BIJ NIET–SPRONGINSTRUCTIES



IND ← PAG, Y . . . Y

ADRES ← MEM [IND] + R [X X]

R [0] ← R [0] + MEM [ADRES]

VOORBEELD:

STEL PAG	=	0 1	= 8.192
STEL Y . . . Y	=	0 0 0 0 0 0 1 0 0 1 0 1 1	= 75
IND = PAG, Y . . . Y	=	0 1 0 0 0 0 0 0 1 0 0 1 0 1 1	= 8.267
STEL MEM [IND]	=	1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1	= 24.583
R [X X]	=	1 0 1 0 1 0 1 0	= 170
ADRES 2e OPERAND	=	1 1 0 0 0 0 0 1 0 1 1 0 0 0 1	= 24.753

Fig. 8

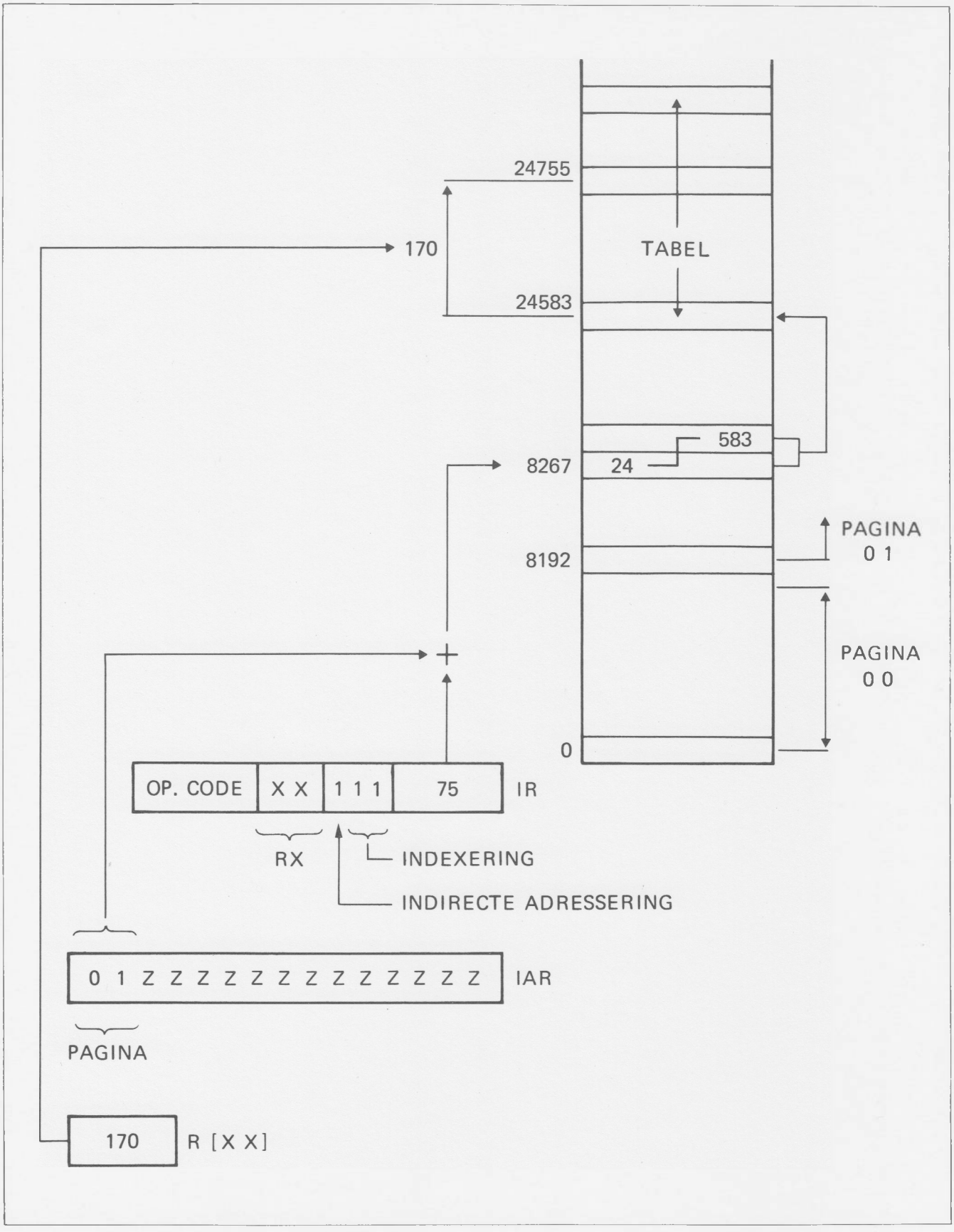

24755
170
TABEL
24583
583
8267
24
PAGINA
0 1
8192
PAGINA
0 0
0
OP. CODE
X X
1 1 1
75
IR
RX
INDEXERING
INDIRECTE ADRESSERING
0 1 Z Z Z Z Z Z Z Z Z Z Z Z Z
IAR
PAGINA
170
R [X X]

Fig. 9

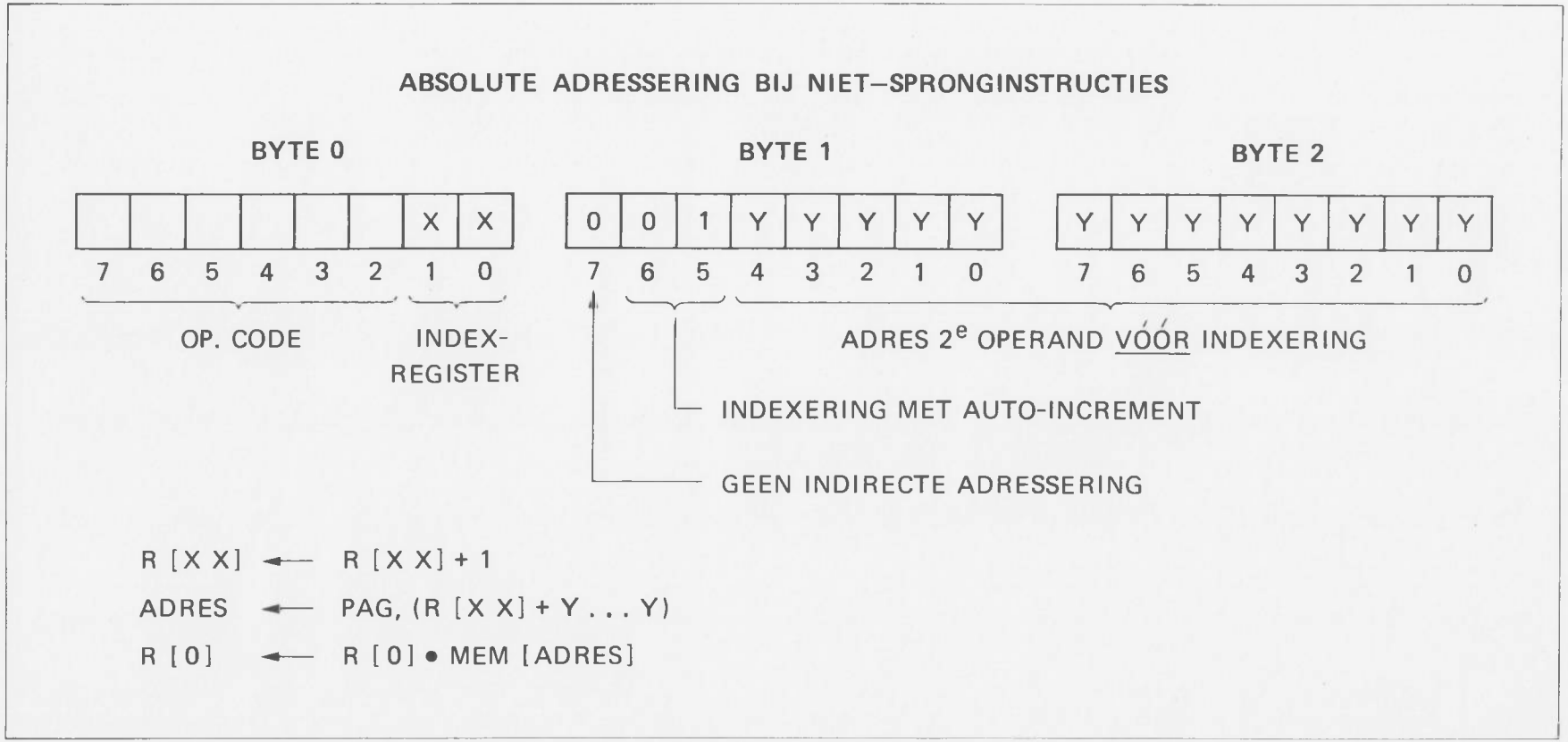

ABSOLUTE ADRESSERING BIJ NIET–SPRONGINSTRUCTIES
BYTE 0
BYTE 1
BYTE 2
X X
0 0 1 Y Y Y Y Y
Y Y Y Y Y Y Y Y
7 6 5 4 3 2 1 0
7 6 5 4 3 2 1 0
7 6 5 4 3 2 1 0
OP. CODE
INDEX-
REGISTER
ADRES 2e OPERAND VÓÓR INDEXERING
INDEXERING MET AUTO-INCREMENT
GEEN INDIRECTE ADRESSERING
R [X X] ← R [X X] + 1
ADRES ← PAG, (R [X X] + Y . . . Y)
R [0] ← R [0] • MEM [ADRES]

Fig. 10

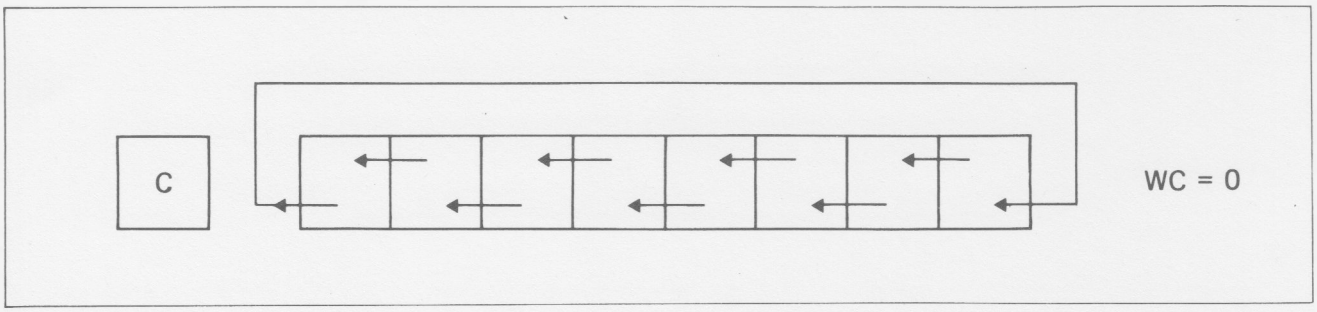


Fig. 11

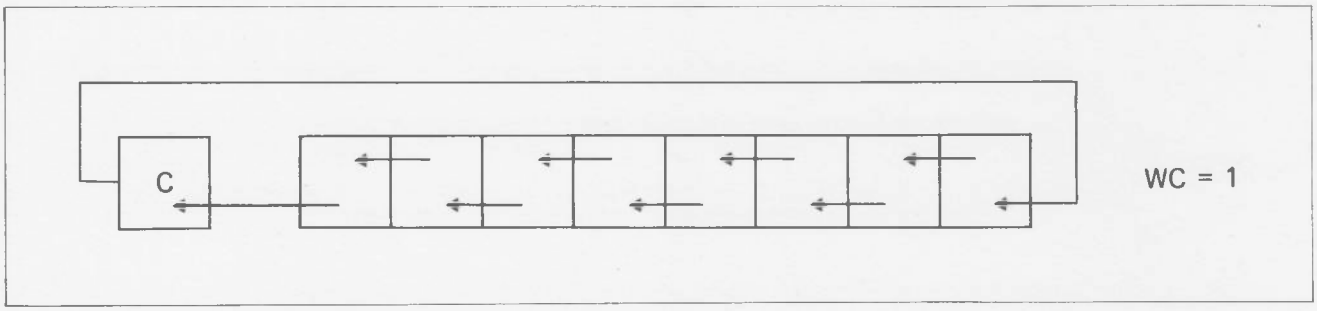


Fig. 12

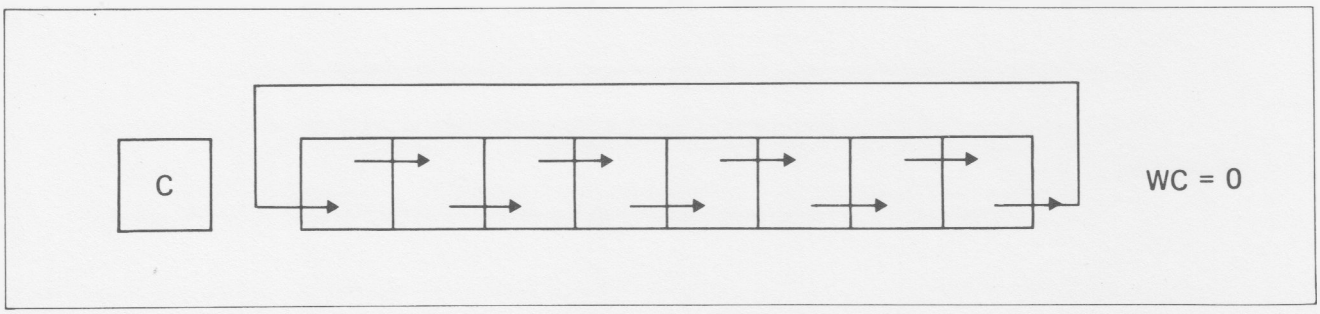


Fig. 13

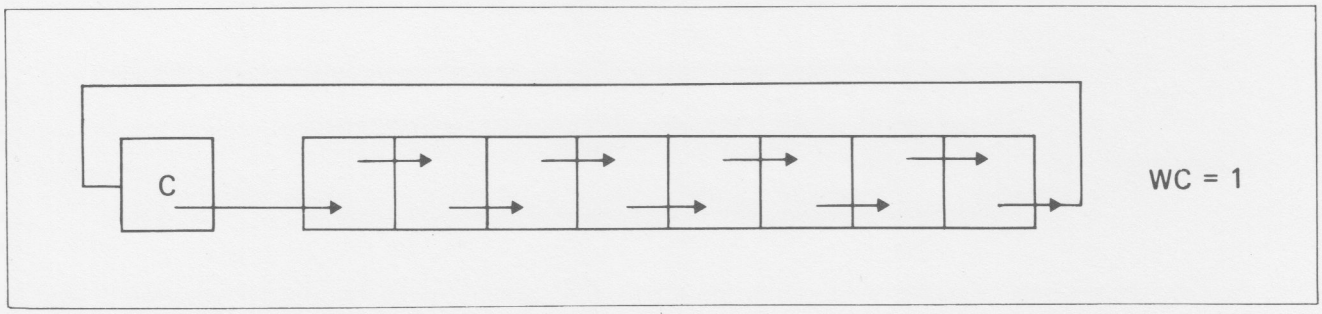


Fig. 14

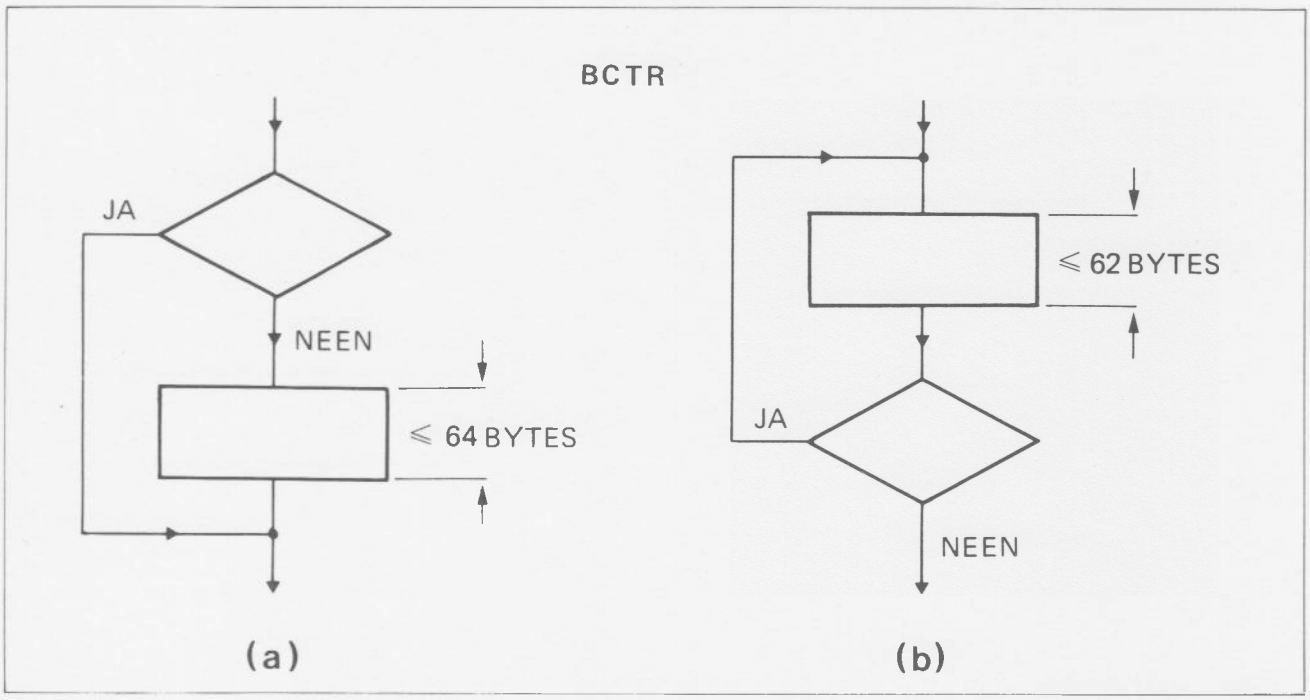


Fig. 15

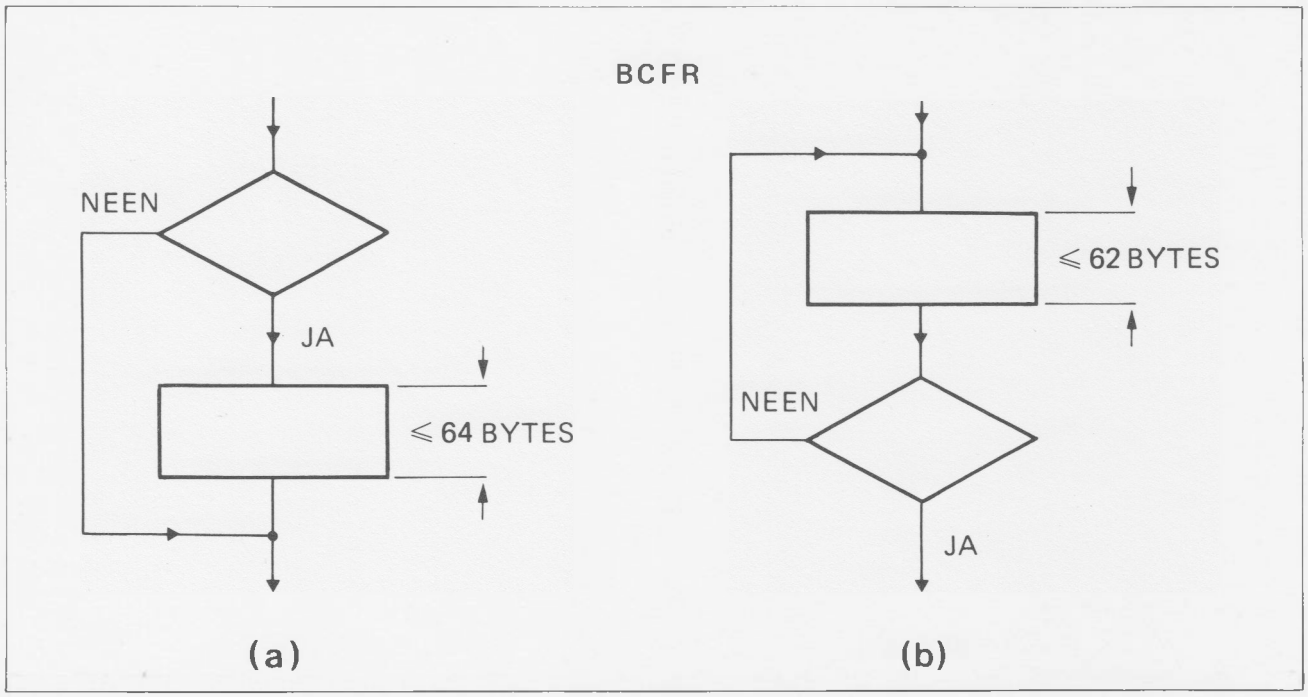


Fig. 16

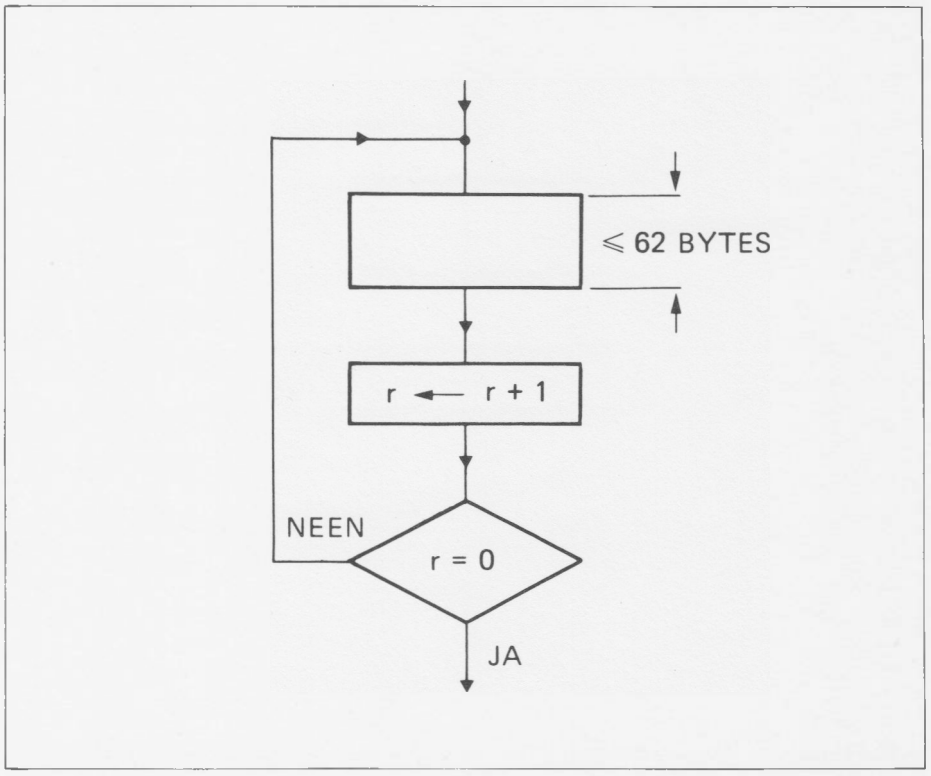


Fig. 17

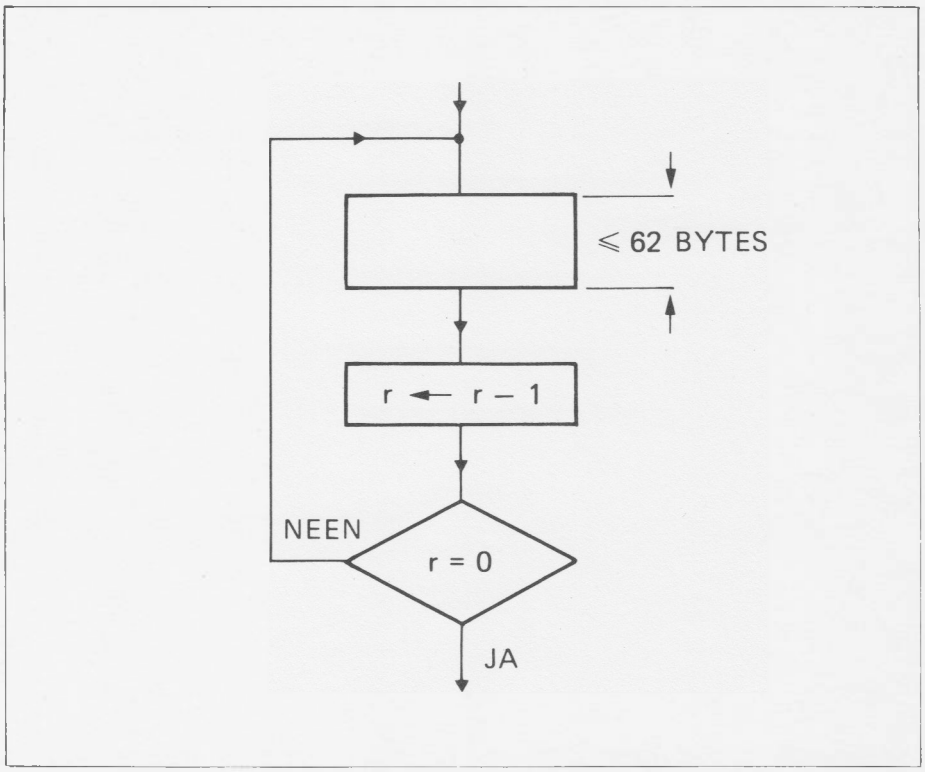


Fig. 18

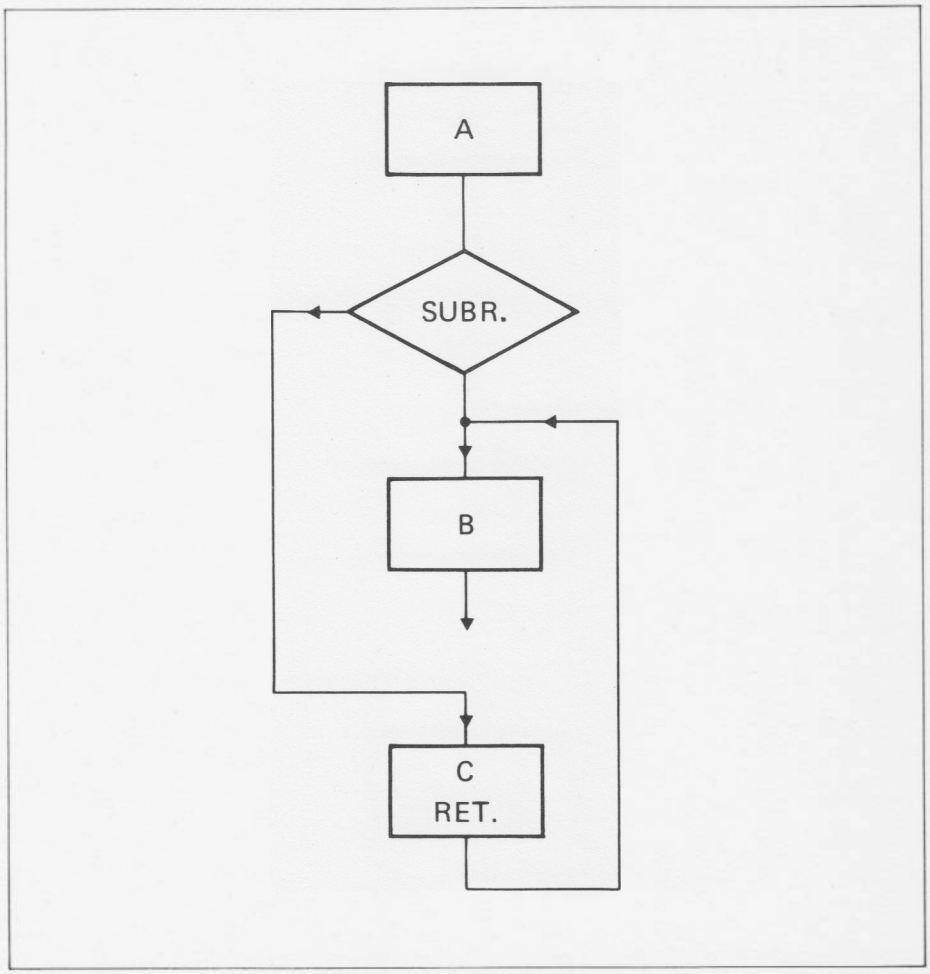


Fig. 19

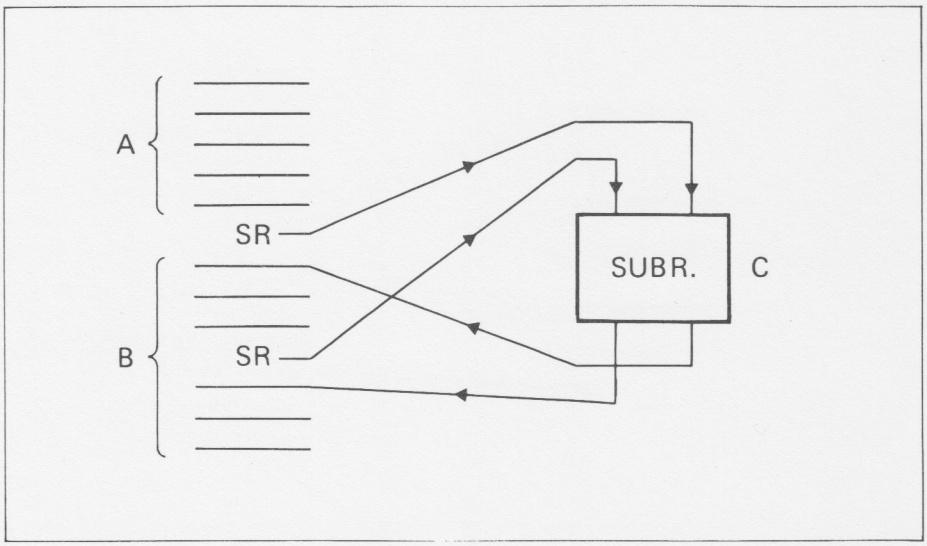


Fig. 20

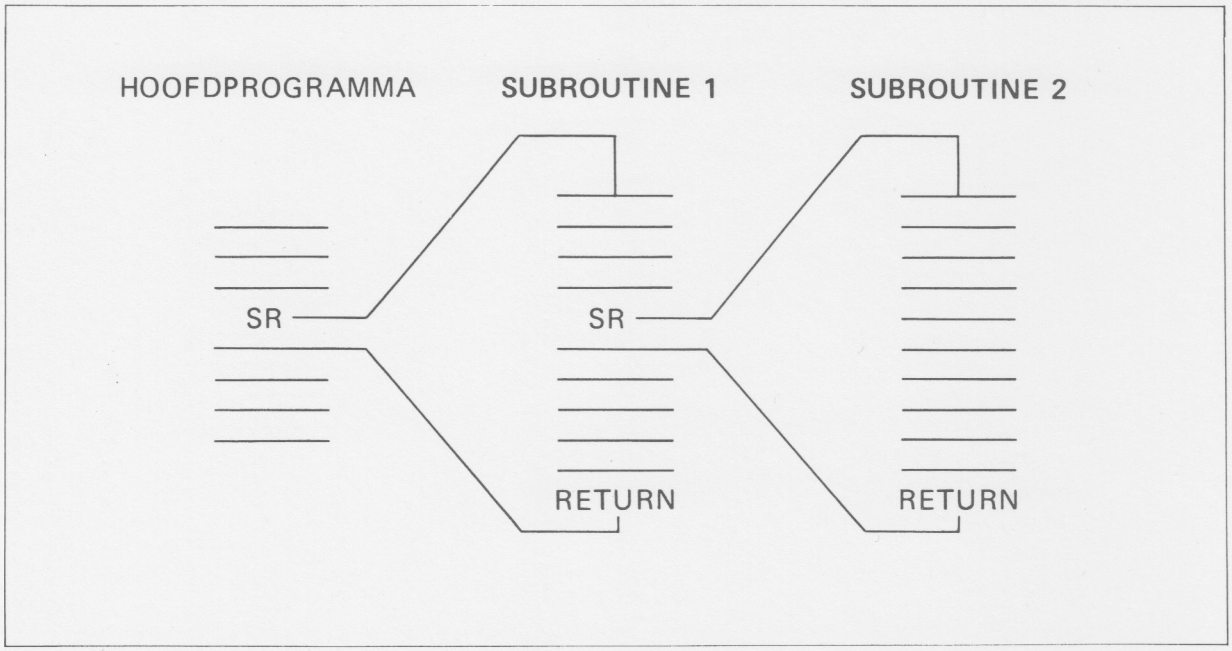

HOOFDPROGRAMMA
SUBROUTINE 1
SUBROUTINE 2
SR
SR
RETURN
RETURN

Fig. 21

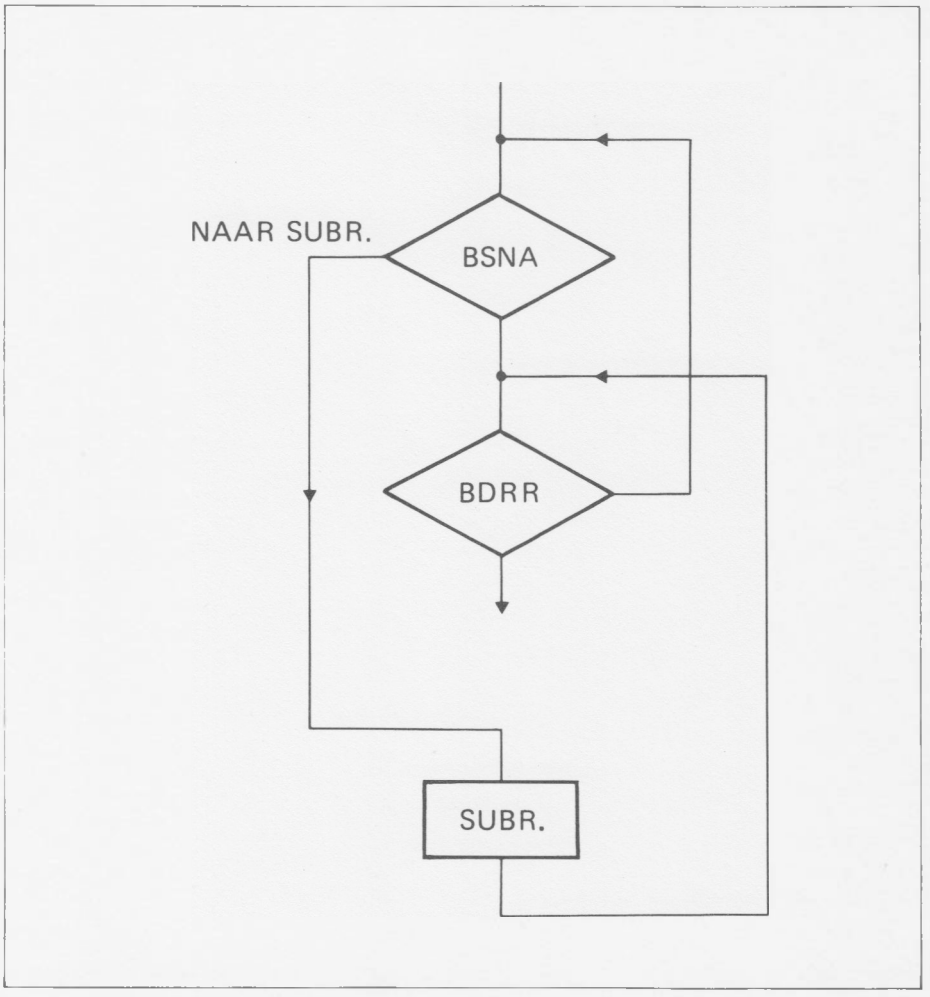


Fig. 22

ZONDER DECIMALE CORRECTIE

STEL A = 75 = H'0111 0101'
B = 36 = H'0011 0110'

(ONJUIST) PROGRAMMA

```
LODA,RO A       RO ← H'75'          0111  0101
ADDA,RO B       RO ← H'36'          0011  0110
                                    ----------
                                RO = 1010  1011 = H'AB'
```

CORRECT PROGRAMMA

```
LODI,RO H'66'   RO ← H'66'          0110  0110
ADDA,RO A       RO ← H'66'+H'75'    0111  0101
                                    ----------  +
                                RO = 1101  1011
ADDA,RO B       RO ← RO+H'36'       0011  0110
                                    ----------  +
                              CY → 1 0001  0001
                                   [1]   [1]
DAR                                 0000  0000
                                    ----------
                                RO  0001  0001 = H'11'
```

STEL A = 75
B = 16

```
LODI,RO H'66'   RO ← H'66'          0110  0110
ADDA,RO A       RO ← H'66'+H'75'    0111  0101
                                    ----------
                                RO = 1101  1011
ADDA,RO B       RO ← RO+H'16'       0001  0110
                                    ----------
                                    1111  0001
                                         [1]
DAR                                 1010  0000
                                    ----------
                                    1001  0001 = H'91'
                                    GEEN CARRY
                                    UITVOEREN
```

Fig. 23

AFTREKKEN MET DECIMALE CORRECTIE

STEL A = 75 = 0111 0101

B = 36 = 0011 0110

PROGRAMMA

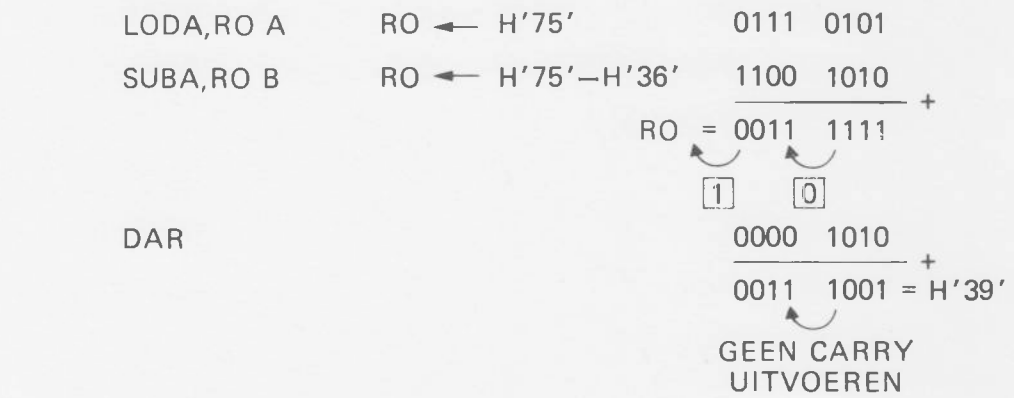


Fig. 24

Electronic components and materials

for professional, industrial and consumer uses

from the world-wide Philips Group of Companies

Argentina: FAPESA I.y.C., Av. Crovara 2550, Tablada, Prov. de BUENOS AIRES, Tel. 652-7438/7478.
Australia: PHILIPS INDUSTRIES HOLDINGS LTD., Elcoma Division, 67 Mars Road, LANE COVE, 2066, N.S.W., Tel. 427 08 88.
Austria: ÖSTERREICHISCHE PHILIPS BAUELEMENTE Industrie G.m.b.H., Triester Str. 64, A-1101 WIEN, Tel. 62 91 11.
Belgium: M.B.L.E., 80, rue des Deux Gares, B-1070 BRUXELLES, Tel. 523 00 00.
Brazil: IBRAPE, Caixa Postal 7383, Av. Brigadeiro Faria Lima, 1735 SAO PAULO, SP, Tel. (011) 211-2600.
Canada: PHILIPS ELECTRONICS LTD., Electron Devices Div., 601 Milner Ave., SCARBOROUGH, Ontario, M1B 1M8, Tel. 292-5161.
Chile: PHILIPS CHILENA S.A., Av. Santa Maria 0760, SANTIAGO, Tel. 39-40 01.
Colombia: SADAPE S.A., P.O. Box 9805, Calle 13, No. 51 + 39, BOGOTA D.E. 1., Tel. 600 600.
Denmark: MINIWATT A/S, Emdrupvej 115A, DK-2400 KØBENHAVN NV., Tel. (01) 69 16 22.
Finland: OY PHILIPS AB, Elcoma Division, Kaivokatu 8, SF-00100 HELSINKI 10, Tel. 1 72 71.
France: R.T.C. LA RADIOTECHNIQUE-COMPELEC, 130 Avenue Ledru Rollin, F-75540 PARIS 11, Tel. 355-44-99.
Germany: VALVO, UB Bauelemente der Philips G.m.b.H., Valvo Haus, Burchardstrasse 19, D-2 HAMBURG 1, Tel. (040) 3296-1.
Greece: PHILIPS S.A. HELLENIQUE, Elcoma Division, 52, Av. Syngrou, ATHENS, Tel. 915 311.
Hong Kong: PHILIPS HONG KONG LTD., Elcoma Div., 15/F Philips Ind. Bldg., 24-28 Kung Yip St., KWAI CHUNG, Tel. NT 24 51 21.
India: PEICO ELECTRONICS & ELECTRICALS LTD., Ramon House, 169 Backbay Reclamation, BOMBAY 400020, Tel. 295144.
Indonesia: P.T. PHILIPS-RALIN ELECTRONICS, Elcoma Division, 'Timah' Building, Jl. Jen. Gatot Subroto, P.O. Box 220, JAKARTA, Tel. 44 163.
Ireland: PHILIPS ELECTRICAL (IRELAND) LTD., Newstead, Clonskeagh, DUBLIN 14, Tel. 69 33 55.
Italy: PHILIPS S.p.A., Sezione Elcoma, Piazza IV Novembre 3, I-20124 MILANO, Tel. 2-6994.
Japan: NIHON PHILIPS CORP., Shuwa Shinagawa Bldg., 26-33 Takanawa 3-chome, Minato-ku, TOKYO (108), Tel. 448-5611.
(IC Products) SIGNETICS JAPAN, LTD, TOKYO, Tel. (03)230-1521.
Korea: PHILIPS ELECTRONICS (KOREA) LTD., Elcoma Div., Philips House, 260-199 Itaewon-dong, Yongsan-ku, C.P.O. Box 3680, SEOUL, Tel. 794-4202.
Malaysia: PHILIPS MALAYSIA SDN. BERHAD, Lot 2, Jalan 222, Section 14, Petaling Jaya, P.O.B. 2163, KUALA LUMPUR, Selangor, Tel. 77 44 11.
Mexico: ELECTRONICA S.A. de C.V., Varsovia No. 36, MEXICO 6, D.F., Tel. 533-11-80.
Netherlands: PHILIPS NEDERLAND B.V., Afd. Elonco, Boschdijk 525, 5600 PB EINDHOVEN, Tel. (040) 79 33 33.
New Zealand: PHILIPS ELECTRICAL IND. LTD., Elcoma Division, 2 Wagener Place, St. Lukes, AUCKLAND, Tel. 867 119.
Norway: NORSK A/S PHILIPS, Electronica, Sørkedalsveien 6, OSLO 3, Tel. 46 38 90.
Peru: CADESA, Rocca de Vergallo 247, LIMA 17, Tel. 62 85 99.
Philippines: PHILIPS INDUSTRIAL DEV. INC., 2246 Pasong Tamo, P.O. Box 911, Makati Comm. Centre, MAKATI-RIZAL 3116, Tel. 86-89-51 to 59.
Portugal: PHILIPS PORTUGESA S.A.R.L., Av. Eng. Duharte Pacheco 6, LISBOA 1, Tel. 68 31 21.
Singapore: PHILIPS PROJECT DEV. (Singapore) PTE LTD., Elcoma Div., P.O.B. 340, Toa Payoh CPO, Lorong 1, Toa Payoh, SINGAPORE 12, Tel. 53 88 11.
South Africa: EDAC (Pty.) Ltd., 3rd Floor Rainer House, Upper Railway Rd. & Ove St., New Doornfontein, JOHANNESBURG 2001, Tel. 614-2362/9.
Spain: COPRESA S.A., Balmes 22, BARCELONA 7, Tel. 301 63 12.
Sweden: A.B. ELCOMA, Lidingövägen 50, S-115 84 STOCKHOLM 27, Tel. 08/67 97 80.
Switzerland: PHILIPS A.G., Elcoma Dept., Allmendstrasse 140-142, CH-8027 ZÜRICH, Tel. 01/43 22 11.
Taiwan: PHILIPS TAIWAN LTD., 3rd Fl., San Min Building, 57-1, Chung Shan N. Rd, Section 2, P.O. Box 22978, TAIPEI, Tel. 5513101-5.
Thailand: PHILIPS ELECTRICAL CO. OF THAILAND LTD., 283 Silom Road, P.O. Box 961, BANGKOK, Tel. 233-6330-9.
Turkey: TÜRK PHILIPS TICARET A.S., EMET Department, Inonu Cad. No. 78-80, ISTANBUL, Tel. 43 59 10.
United Kingdom: MULLARD LTD., Mullard House, Torrington Place, LONDON WC1E 7HD, Tel. 01-580 6633.
United States: (Active devices & Materials) AMPEREX SALES CORP., Providence Pike, SLATERSVILLE, R.I. 02876, Tel. (401) 762-9000.
(Passive devices) MEPCO/ELECTRA INC., Columbia Rd., MORRISTOWN, N.J. 07960, Tel. (201) 539-2000.
(IC Products) SIGNETICS CORPORATION, 811 East Arques Avenue, SUNNYVALE, California 94086, Tel. (408) 739-7700.
Uruguay: LUZILECTRON S.A., Rondeau 1567, piso 5, MONTEVIDEO, Tel. 9 43 21.
Venezuela: IND. VENEZOLANAS PHILIPS S.A., Elcoma Dept., A. Ppal de los Ruices, Edif. Centro Colgate, CARACAS, Tel. 36 05 11.

A15

Printed in The Netherlands

9398 6082 0221