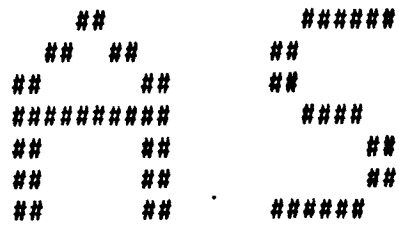

Conditionele macro assembler voor de 65(C)02 microprocessor.

A.B.M. Brouwer
Nieuwkoopseweg 44
2641 PB Pijnacker

E.J.M. Visschedijk
Drakensteyn 299
7608 TR Almelo

Inhoudsopgave

Inleiding

AS is een speciaal voor DOS65 ontworpen conditionele macro assembler. Met de screeneditor worden de sourcefiles gegenereerd en AS assembleert deze. De sourcefiles mogen wat lengte betreft vele malen groter zijn dan het geheugen. Zelfs de gegenereerde object hoeft niet in het geheugen te passen. Deze wordt namelijk weer op schijf weggeschreven. Alleen de label tabel wordt in het geheugen gezet. Het is mogelijk libraries aan de sourcefiles te linken. Eveneens is het mogelijk locale en globale labels te gebruiken. In AS kunnen tevens uitgebreide rekenkundige bewerkingen gedaan worden.

1. Syntax

De assembler wordt gestart met het commando 'as filenaam'. AS zal de source file lezen en hieruit direct uitvoerbare 65(C)02 code genereren.

De source filenamen hebben default de extensie .mac. De object file krijgt default de extensie .bin. Dat houdt tevens in dat AS binary files genereert. Binary files onderscheiden zich van files die met bv. het commando SAVE op schijf worden gezet. Bij het commando SAVE (dan wel zonder de optie -B!!!) wordt het laadadres van de file, evenals het startadres in de header van de file gezet. Verder heeft zo'n file een data gedeelte. Dat data gedeelte wordt aaneengesloten in het geheugen gezet mbv. LOAD. AS genereert echter een file waarin laadadres en startadres in het data gedeelte staan. Deze files kunnen met LOAD in het geheugen gezet worden. Als laadadres worden de adressen genomen die in de file staan. Het commando LOAD gevolgd door een laadadres werkt niet!! Dan wordt namelijk de laadadres- en startadres informatie ook in het geheugen gezet. Wil men een binary file toch verschoven in het geheugen laden, dan gebruike men OLOAD. Aan het eind van deze manual volgt een voorbeeld. Hierin staat beschreven hoe met een door AS gegenereerde binary file moet worden omgesprongen.

As assembleert de source in twee fasen, pass 1 en pass 2. Alleen tijdens pass 2 komen er foutmeldingen op het scherm en wordt er code naar de object file geschreven.

Opties

Opties mogen voor of achter de filenaam worden opgegeven. Opties die gevolgd worden door een string of getal moeten afgesloten worden met een "witte" spatie (spatie, tab of einde regel). Losstaande opties mogen achter elkaar worden opgegeven.

-l	laat een listing zien. Een 'opt nolis' instruktie stopt het printen tot een 'opt lis' instruktie.
-s	print na afloop de globale labels.
-n	print de source regels met regelnummers voor het labelveld.
-o	geen binary object file.
-cnnn	definieer de maximale lengte van de listing regel. Maximaal is dit 240, minimaal 10. Default wordt 240 genomen, optie -c alleen definieert de lengte op 78.
-aargument,..	definieer een argument voor de argm instruktie. Deze argumenten moeten in hetzelfde formaat worden opgegeven als een standaard macro aanroep. In het argument kunnen echter geen spaties voorkomen.
-bbinfile	definieer de binary object filenaam. De file krijgt default extensie '.bin'.

'As -lns test' assembleert 'test.mac' met object file 'test.bin', met listing, regelnummers en listing label tabel.
'As -boutput.obj -c -l test.txt' assembleert 'test.txt' met binary object file 'output.obj' en listing waarbij de regels tot 78 karakters beperkt worden (terminal breedte).
'As test -lc96 -o' assembleert test zonder binary output file, met listing waarbij de regel beperkt blijft tot 96 karakters.
'As test -a"text",3' assembleert 'test.mac' met binary output file, met als argumenten '"text"' en '3'.

2. Source formaat

Een regel source heeft vier velden, n.l. label instruktie operand en een commentaar veld. Als een instruktie geen operand nodig heeft, dan wordt eventueel volgende tekst als commentaar opgevat.

De velden worden van elkaar gescheiden door een of meer "witte" spaties. Dit geldt niet voor het operand veld wanneer spaties zich binnen een string bevinden of als er een karakter als konstante gedefinieerd wordt.

In elk veld kan een commentaar veld gestart worden d.m.v. het ';' karakter. Lege regels zijn ook commentaar regels.

```
label   inx                 voorbeeld
label   sta     label       voorbeeld
label                       ;voorbeeld ';' noodzakelijk omdat
;                           anders 'voorbeeld' als instruktie
;                           wordt beschouwd.
```

Labelvelden mogen 40 karakters lang zijn, instruktievelden 20 karakters en afzonderlijke operands in het operandveld 80 karakters. De totale regel mag 160 karakters lang zijn. Deze afmetingen zijn gekozen rekening houdend met het feit dat de originele source velden aanzienlijk langer kunnen worden tijdens macro executie wanneer parameters door argumenten worden vervangen.

3. 65(C)02 Instruktie set

Standaard 6502 instrukties. Zover mogelijk en wenselijk is de standaard mnemonic gehandhaaft.

adc	and	asl	asla	bcc	bcs	beq	bit
bmi	bne	bpl	brk	bvc	bvs	clc	cld
cli	clv	cmp	cpx	cpy	dec	dex	dey
eor	inc	inx	iny	jmp	jsr	lda	ldx
ldy	lsr	lsra	nop	ora	pha	php	pla
plp	rol	rola	ror	rora	rti	rts	sbc
sec	sed	sei	sta	stx	sty	tax	tay
tsx	txa	txs	tya				

Standaard 65C02.
Deze wordt geselecteerd met de pseudo instruktie 'opt c02'. Naast een aantal standaard instrukties waarvan de adresseringsmogelijkheden zijn uitgebreid, zijn er ook nieuwe instrukties:

bra	deca	inca	stz	trb	tsb	phx	phy
plx	ply						

Rockwell 65C02.
Deze worden tesamen met de andere 65C02 uitbreidingen geselecteerd met de pseudo instruktie 'opt rc02'. De volgende instrukties zijn toegevoegd:

bbr	bbs	rmb	smb

4. Instruktie operand

Instrukties zonder operand:
6502:

asla	brk	clc	cld	cli	clv	dex	dey
inx	iny	lsra	nop	pha	php	pla	plp
rola	rora	rti	rts	sec	sed	sei	tax
tay	tsx	txa	txs	tya			

65C02:

bra	deca	inca	phx	phy	plx	ply

De andere instrukties hebben een of meer operands.

De vier rockwell 65C02 instrukties hebben twee of drie operands. Er is gekozen om het bit nummer dat bij deze instrukties hoort, naar het operand veld te verplaatsen omdat het dan mogelijk is het bit nummer d.m.v. een label voor te stellen. De syntax is als volgt:

```
bbr (bbs)       #bit,memoryadres,sprongadres
rmb (smb)       #bit,memoryadres
```

5. Addresserings keuze

<opr forced page zero. As kiest default page zero adressering als de operand kleiner is dan 256 en de page zero adresseringsmode voor die instruktie aanwezig is.

>opr forced absolute. Dit kan nodig zijn in het geval van opr,x adressering, waarbij het effectieve adres hoger is dan 255.

#opr immediate.

opr,x indexed x.
opr,y indexed y.

[opr] indirect.
[opr,x] indexed x indirect.
[opr],y indirect indexed y.

6. Pseudo instrukties

	argl	argument	definieer een default argument in het geval de gebruiker geen -a argument meegeeft.
label	argm	parameter,..	definieer de argument macro. Parameters die gedefinieerd zijn worden vervangen door het argl argument. Label is de naam die aan het argument toegekend wordt (dit label wordt verder niet gebruikt).
	cont	filenaam	sluit de huidige source file en ga verder met file filenaam.
	else		assembleer tot endif indien er binnen de huidige if-endif nog niet geassembleerd is.
	elseif	conditie	assembleer tot de volgende endif, elseif of else indien er binnen de huidige if-endif nog niet geassembleerd is en conditie waar is.
	end	[adres]	stop met lezen uit de huidige file. Indien adres wordt opgegeven, dan wordt dit als startadres naar de object file verstuurd. Dit gebeurt niet voor lib files.
	endif		sluit het huidige if-endif blok.
	endm		einde macro definitie.
label	equ	expressie	definieer een label met waarde expressie.
	exitm		verlaat de huidige macro.
[label]	fcb	expressie,..	genereer bytes expressie.
[label]	fcc	expressie,..	genereer bytes gelijk fcb. Expressie mag hierbij ook een string zijn omsloten door ''' of '"'.
[label]	fdb	expressie,..	genereer woorden expressie.
	if	expressie	assembleer de volgende source tot endif, elseif of endif indien expressie waar is.
	ifb	expressie,offset	spring binnen de huidige macro indien expressie waar is. Offset gelijk aan 1 slaat de volgende regel over, offset -1 springt naar de vorige macro regel.

	lib	filenaam	voeg de source van file filenaam tussen die van de huidige file. Dit kan 2 niveaus diep.
label	macro	parameter,..	definieer de volgende source regels tot endm als macro met naam label.
	opt	optie,..	(re)set assembler opties. Behalve optie con staan alle opties aan. Zie het speciale hoofdstuk over de opt instruktie.
[label]	org	expressie	zet de programmateller op expressie.
	pag		begin de volgende regel op een nieuwe bladzijde. Er wordt op elke pagina een kop geprint indien optie pag aan is en de titel (ttl) string gedefinieerd is. Een pagina wordt afgesloten door een form-feed ($0c).
	pag	expressie	definieer de pagina lengte (default 58). Indien expressie negatief is, wordt er op een nieuwe pagina begonnen als er minder dan -(expressie) regels op de huidige bladzijde over zijn.
[label]	res	expressie	reserveer een stuk geheugen ter grootte van expressie bytes.
[label]	res	expressie,data	vul expressie bytes met waarde data.
label	set	expressie	geef label de waarde expressie. Zonodig wordt tijdens pass 1 dit label gecreeerd.
	ttl	string	definieer een kop regel die boven elke bladzijde wordt afgedrukt. Zie het speciale hoofdstuk over de ttl instruktie.

7. Labels

Karakters die een label vormen zijn:

A-Z, a-z, 0-9, _, ., en $.

Als het eerste karakter van een label een nummer is, dan is dit een locaal label. Volgende karakters mogen alleen nummers zijn. De waarde van dit getal mag liggen tussen 0 en 255 (Grotere getallen worden modulo 256 genomen).

Locale labels zijn niet toegestaan als 'EQU' of 'SET' label.

Labels mogen tijdens declaratie worden afgeloten met ':' of met '.' voor locale labels.

Globale labels mogen 40 karakters lang zijn. Kleine en hoofdletters zijn verschillend.

8. Expressies

Een expressie mag bestaan uit labels of getallen met daartussen rekenkundige, relationele en/of logische operaties.

Een label kenmerkt zich met een niet nummer en geen '$' als eerste karakter (dus A-Z, a-z, _, of .). Een decimaal getal gevolgd door '.' is een locaal label. Locale labels kunnen niet uniek zijn. Om het zoeken te dwingen is het karakter achter de '.' het dwing karakter. Deze karakters zijn:

F,f zoek het label dat het kleinste positieve verschil ongelijk nul heeft met de huidige programmateller (ligt "verder").

B,b zoek het label dat het kleinste negatieve verschil heeft met de huidige programmateller (ligt "terug").

Zonder dit dwing karakter kiest AS automatisch het label dat absoluut gezien de kleinste afstand heeft tot de waarde van de huidige programma teller. Als de afstand tot het vorige label gelijk is aan de afstand tot het volgende label dan kiest AS voor het volgende label.

Getallen zijn er in decimaal, binair, octaal en hexadecimaal. Het talstelsel wordt gedefinieerd door het eerste of het laatste karakter. Het talstelsel is decimaal als er geen talstelsel wordt gespecificeerd.

%	%0001	binair
@	@377	octaal
$	$a05	hexadecimaal
..B	101b	binair
..O	563o	octaal
..H	08fh	hexadecimaal

Constanten

$	$+1	huidige programmateller. Het '$' karakter wordt als zodanig herkend omdat het niet wordt gevolgd door een karakter dat tot een label behoort.
'.'	'a'	Een (1) karakter. De sluit ''' is niet noodzakelijk maar het gebruik ervan wordt wel aanbevolen.
".."	"cd"	Dubbel karakter (woord). Het sluit karakter '"' hierbij is niet noodzakelijk.

Karakters in een string of als konstante worden direct overgenomen. Het '\' karakter vervult hierbij een speciale functie. Het karakter achter '\' wordt omgezet tot:

\a	$07	bell
\b	$08	backspace
\t	$09	tab
\n	$0a	line feed
\f	$0c	form feed
\r	$0d	carriage return
\E	$1b	escape

\nnn	een getal achter '\' wordt octaal omgezet ('\0' is $00).

Alle andere karakters worden normaal overgenomen ('\\' wordt dus '\').

Operaties

Enkelvoudige operaties werken op de expressie direct achter de operator. Deze operaties zijn:

~	~1	one's complement (bit inversie)
!	!0	logical not (nul wordt een, niet nul wordt een)
-	-5	minus (two's complement)

Meervoudige operaties zijn onder te verdelen in aritmetische, logische, relationele en conditionele operaties.

Aritmetische operaties:

*	3*3	vermenigvuldigen
/	5/3	delen
%	4%3	modulo
+	4+6	optellen
-	4-1	aftrekken
<<	1<<8	schuif naar links (1 acht naar links)
>>	256>>8	schuif naar rechts
&	4&1	en (bit en)
^	9^2	exclusive or
!	1!4	inclusive or

Logische operaties:

==	3==2	gelijk
!=	4!=4	ongelijk
	"de"=="de"	Alleen bij deze twee operaties gelden de operatoren tevens voor strings. In alle andere gevallen is het resultaat niet gedefinieerd. Strings worden alleen herkent bij de fcc of conditionele pseudo instrukties.
<	4<3	kleiner dan
<=		kleiner dan of gelijk
>		groter dan
>=		groter dan of gelijk
&&	3&&0	logische en
!!	3!!0	logische or

Conditionele operatie:

? :	1?2:3	indien dan anders. Bij deze operatie zijn dus twee vervolg operatoren gescheiden door ':' nodig.

De operaties worden van links naar rechts geevalueerd daarbij rekening houdend met de volgende regels:

Een expressie tussen haakjes '( )' heeft de hoogste prioriteit.

In tabelvorm aflopend in prioriteit.

```
-       ~  !  -
-       *  /  %
-       +  -
-       << >>
-       < <= > >=
-       == !=
-       &
-       ^
-       |
-       &&
-       ||
-       ?:
```

Een veel voorkomende expressie is &255 en >>8. Deze wordt gebruikt om een van de registers die 8 bits breed zijn te laden met het low of high byte van een 16 bits breed adres. Willen we de accu laden met het high byte van adres LABEL dan schrijven we:

```
        LDA     #LABEL>>8
```

Het linkse byte is het high byte en moet dus 8 keer naar rechts geschoven worden om een 8 bits getal te leveren. Willen we de accu met het low byte van adres LABEL laden, dan schrijven we:

```
        LDA     #LABEL&255
```

Op het gehele 16 bits getal wordt de AND funktie losgelaten. Er wordt ge-AND met 255 (decimaal voor $FF). Op deze manier wordt het linkse gedeelte van het 16 bits getal op $00 gezet.

9. Conditionele instrukties

De conditionele instrukties if, elseif, else en endif kunnen 10 niveaus diep gaan. Het is dus mogelijk om if-endif konstrukties in elkaar te plaatsen.

Voor if en elseif zijn onbekende labels gelijk aan nul (0) wat het conditioneel assembleren beperkt tot het wel of niet van te voren definieeren van dat label ongelijk aan nul.

Het is verder mogelijk om een conditie afhankelijk van het eerste karakter van het label te laten zijn. Dit gebeurt in de vorm van 'if karakter,operand'. Voor karakters die toegelaten worden als label (bv. 'a') geldt dan dat die operand exact gelijk moet zijn aan het karakter waarmee het vergeleken wordt (bv. 'a,a'). Voor andere karakters is het voldoende dat het eerste karakter gelijk is (bv. '#,#5').

Met de '.,operand' expressie kan getest worden of een operand aanwezig is. De expressie is waar in het geval dat er geen operand is.

Tijdens het verwerken van operands als '>#5' worden er karakters overgeslagen zolang deze behoren tot '<', '>' of '#'. Deze karakters worden voor standaard instrukties gebruikt om de assembler voor een bepaald type operand te laten kiezen. Voor pseudo instrukties (bv. fcc) zijn ze van geen nut en worden dan overgeslagen.

De gegeven opzet is voornamelijk van belang voor macro's. Door macro's zorgvuldig te definieren, kan met een macro definitie gekozen worden uit verschillende uitvoer.

10. Macro's

Macro's kunnen tot 10 niveaus diep gaan. Zij kunnen daarbij andere macro's aanroepen waarbij dan de eventueel bijbehorende argumenten doorgegeven worden.

Macro definities mogen niet langer zijn dan 255 regels.

Labels, instrukties of operands worden door meegegeven argumenten vervangen indien het label, de instruktie of operand gelijk is aan een parameter in de macro definitie.

Parameters in de macro definitie worden gescheiden door ','. Alle karakters worden opgenomen in de lijst maar er volgt een foutmelding indien er een niet label karakter in een parameter voorkomt.

Bij de aanroep worden de argumenten gescheiden door ','. Als het '\' karakter de ',' voorgaat, dan wordt het ',' karakter opgenomen in het argument. Hiermee is het mogelijk om gehele operands over te nemen (bv 'temp\,x' in de aanroep wordt 'temp,x' als operand tijdens macro expansie).

Het argument wordt beschouwd als string indien dit omsloten wordt met ''' of '"'. Indien dit karakter voorafgegaan wordt door '\', dan geldt dit karakter niet als openings of sluitings teken maar wordt dan letterlijk overgenomen. Indien het de bedoeling is een (1) of een dubbel karakter mee te geven, dan moet dit afgesloten worden (voor niet macro's was dit niet verplicht).

11. TTL Instruktie

De operand van de ttl instuktie moet als string opgegeven worden. De eerste drie karakters van de string bepalen de argumenten die meegegeven worden. Dit zijn:

t	datum en tijd.
f	huidige filenaam.
p	bladzijde nummer.

Indien voor het vierde karakter een karakter staat dat niet voldoet aan de hiervoor genoemde letters, dan wordt dit scannen gestopt en wordt er verder gegaan met het printen van de string.

Uitvoer controle karakters.

%d	print het volgende argument decimaal (pagina nummer).
%b	print het volgende argument binair.
%o	print het volgende argument octaal.
%x	print het volgende argument hexadecimaal.
%c	print het volgende argument als karakter.
%s	print het volgende argument als string (tijd datum, filenaam).
	elk ander karakter wordt geprint (dus '%%' print '%').

Een decimaal nummer tussen '%' en het format karakter plaatst het uitvoer argument in een n breed veld. Dit wordt dan aan de linker kant opgevuld met spaties of '0' indien het eerste karakter van het nummer '0' is. Het argument wordt aan de linker kant van het veld geplaatst als voor het nummer het '-' karakter staat.

Als het nummer gevolgd wordt door '.' en een nummer, dan wordt het argument maximaal dat aantal breed.

vb. "t.Dit is de tijd %s" Dit is de tijd 13-Nov-85 10:21

Dit laat de tijd zien. De '.' achter 't' stopt het plaatsen van argumenten maar wordt zelf niet afgedrukt.

```
"tfpTijd %s  file %s  pagina %d"
Tijd 13-Nov-85 10:21  file TEST.MAC  pagina 1
```

Hieruit is te zien dat 'tfp' in dezelfde volgorde moet staan als de volgorde waarin ze geprint worden. Het verkeerd opgeven heeft meestal vreemde gevolgen.

```
"tfpTijd %s  file %15s  pagina %04d"
Tijd 13-Nov-85 10:21  file          TEST.MAC  pagina 0001

"fp.File %-20s  blz. %4d"
File TEST.MAC             blz.    1

"fp.File %-20.4s  blz. %4d"
File TEST                 blz.    1
```

Omdat het argument van ttl eerst door de string processor gaat, worden backslashes eerst vertaald tot speciale karakters.

".\7" als ttl string print het bel karakter op elke pagina.

12. OPT Instruktie

De mogelijke opties zijn:

(no)lis	list control. Er wordt alleen een listing gegenereerd indien de assembler met de -l optie is gestart.
(no)pag	tel de regels per pagina. Print de titel regel boven elke pagina en sluit de pagina af met een form-feed.
(no)gen	meervoudige uitvoer regels voor fcb, fcc en fdb indien dit nodig is.
(no)mac	laat de macro aanroep regel zien. Deze wordt tevens geprint indien er een label voor de macro instruktie staat.
(no)exp	print regels tijdens macro uitvoering.
(no)con	print niet geassembleerd source (conditionele source).
(no)mne	laat pseudo instrukties regels zien die alleen dienen voor de assembler. Als een label optioneel is worden zij altijd geprint indien er een label voor de instruktie staat.
c02	selecteer bij de standaard 6502 instrukties de 65C02 instrukties en toegevoegde adresserings mogelijkheden.
rc02	selecteer bij de 65C02 de rockwell 65C02 instrukties.

13. Assembler foutmeldingen

phasing error detected
Deze melding wordt gegeven als gezien wordt dat het programma er tijdens pass 2 anders uitziet dan tijdens pass 1. Dit geldt niet voor de operand velden waar de labels vaak pas bekend zijn na pass 1 maar voor de instrukties die op een andere plaats terecht komen. Deze positiefout wordt herkend op een regel met een globaal label dat de waarde krijgt (pass 1) of moet hebben (pass 2) van de huidige programma teller. De positie fout is dus opgetreden tussen het huidige globale label en de regel met het vorige globale programmateller label.
Dit probleem wordt meestal veroorzaakt door een operand die in pass 2 een zodanig verschil heeft met dezelfde operand tijdens pass 1 dat de assembler tijdens pass 2 niet dezelfde adressering toepast als tijdens pass 1. Uit voorzorg hiervoor neemt AS tijdens pass 1 aan dat een onbekend label later een waarde krijgt dat groter is dan 255.
De positie fout kan ook optreden bij conditioneel assembleren als er verschillende gedeeltes geassembleerd worden.

unrecognizable mnemonic or macro
Er is een onbekende instruktie gedefinieerd. Meestal is dit een invoerfout. AS genereert drie NOP instrukties zodat de gebruiker deze desgewenst zelf kan invullen.

operand syntax error
Leeg of niet correct operand veld. De assembler zal in veel gevallen toch een correcte uitvoer geven.

invalid addressng mode
Niet toegestane adresserings mogelijkheid. De assembler zal afhankelijk van de bestaande mogelijkheden en opgegeven adresserings informatie een bestaande instruktie genereren.

missing ']'
Bij de indirecte adressering wordt de sluit ']' gemist. Afhankelijk van de indexering genereerd AS een indirecte instruktie.

branch too long
De relatieve sprong van de branch instruktie is groter dan +127 of kleiner dan -128. Dit geldt ook voor de rockwell 65c02 test en spring instruktie.

operand not at page zero
Het operand adres is groter dan 255. Deze fout wordt gegeven voor de indirecte adressering (behalve jmp) en voor de rockwell 65c02 bit instrukties.

operand truncated
Deze waarschuwing wordt alleen op verzoek (commando optie -w) geprint als een byte operand groter is dan 255. (immediate adresering, fcb instruktie en andere enkelbyte operands).

can't open source file
Voor lib of cont instrukties. Bij de lib instruktie gaat de assembler verder met de huidige file, bij de cont instruktie in library files wordt de fout beschouwd als end-of-file, anders (hoofdprogramma) springt AS, na een melding, naar het operating system.

macro label equals mnemonic
Er wordt een macronaam bij aanvang van een macro definitie gebruikt die al bestaat als standaard opcode. De macro definitie wordt overgeslagen.

macro label missing
Er is geen macronaam in het labelveld bij aanvang van een macro definitie. De macro definitie wordt overgeslagen.

macro not defined in pass 1
Tijdens pass 2 wordt een macro definitie gezien die niet gedefinieerd is tijdens pass 1.

multiple defined macro
Deze macronaam is al in gebruik. De macro definitie wordt overgeslagen.

can't replace parameter
Tijdens macro executie kan een parameter niet vervangen worden door een argument die aan de macro is meegegeven. Meestal zijn er dan bij de aanroep geen of te weinig argumenten meegegeven.

not allowed in this context
Een instruktie die alleen geldig is binnen een macro (exitm) of tijdens een macro definitie (endm) of die niet voor mag komen in een macro (lib).

invalid expression
Een operand expressie die niet klopt. Voor de opt instruktie betekent het dat de gespecificeerde optie niet bestaat (meestal syntax fout).

missing ')'
De expressie die tussen haakjes staat, mist de sluit ')'.

digit not in specified range
In de expressie komt een karakter voor die niet in het bereik ligt van het talstelsel.

label not defined in pass 1
Tijdens pass 2 komt een label voor dat niet tijdens pass 1 gedefinieerd is.

undefined label
Het opgegeven label in het operandveld niet gevonden worden.

missing label
Geen labelnaam in het labelveld bij een equ of set instruktie. De instruktie wordt genegeerd.

invalid label
Een niet correct gedefinieerd label. De labelnaam wordt zover mogelijk gebruikt.

multiple defined label
Het opgegeven label bestaat al. Het huidige label wordt genegeerd. In expressie evaluatie wordt dus alleen het eerste label gebruikt.

unbalanced conditional
Als de conditionele instruktie hier niet mogelijk is. B.v. else of endif zonder dat er eerst een if instruktie aan vooraf gegaan is. Het eenmaal misgaan van een conditionele instruktie (b.v. door er een syntax fout) kan grote gevolgen hebben.

Andere foutmeldingen

Deze foutmeldingen zijn fataal, d.w.z. er wordt direct daarna teruggekeerd naar het operating system. De meeste meldingen laten zien waar in de file de fout optrad.

can't open source file
De file die geassembleerd moet worden, bestaat niet of kan niet geopend worden.

can't create object file
Er kan om de een of andere reden geen object file gecreeerd worden.

error writing object file
Tijdens het schrijven naar de object file komt er een fout voor. Waarschijnlijk is de disk beschermd of vol.

library nesting too deep
Het niveau van library files wordt door deze lib instruktie groter dan 2. Dit aantal is zo gekozen dat het voor normaal gebruik voldoende is.

can't continue
De file die het vervolg is in het hoofdprogramma bestaat niet of kan niet geopend worden. Een continue error voor library files wordt beschouwd als einde file.

if level too deep
Het niveau van conditionele instrukties wordt door deze if instruktie hoger dan 10. Oorzaak kan zijn een if-endif konstruktie in een macro die niet goed gedefinieerd is.

macro nesting too deep
Het niveau van macro's die elkaar aanroepen wordt hoger dan 10.

macro line buffer overflow
De lengte van argumenten van alle actieve macro's is groter dan 200 karakters. Mocht dit probleem zich voordoen, dan kan een oplossing zijn het vervangen van complexe expressies door labels die staan voor de waarde van die expressies.

macro linecount overflow
Het aantal regels in de macro is groter dan 255. Opsplitsing van de macro in meerdere macro's en/of verwijderen van kommentaarregels is de oplossing.

origin specification
In de definitie van de programmateller kwam een of andere fout voor (voor pass 1 betekent dit o.a. dat er geen onbekend label in de expressie mag voorkomen).

out of memory
Er is niet genoeg geheugen of labels en macro's in het geheugen op te slaan. Oplossingen:
- vervang globale labels door locale. Globale labels nemen minimaal 9 bytes in beslag tegen 3 bytes voor locale labels.
- verwijder kommentaar uit macro definities.
- inkorten van de sourceregels in de macrodefinitie. Dit kan zonder veel problemen gebeuren voor de macro parameters.

14. Voorbeelden

Hieronder staan een aantal voorbeelden gegeven hoe vanuit het niets een programma in AS geschreven kan worden. Verder wordt beschreven hoe AS moet worden opgestart en wat de resultaten zijn. Tevens wordt van een met AS gemaakte object file een commando gemaakt.

Stel we hebben in een directory de volgende files staan:

- library.mac — Library file met definities
- prtstr.mac — Eigenlijke programma dat geassembleerd moet worden.
- asprt — Command file die de file prtstr.mac assembleert.

De inhoud van library.mac is.:

```
; file  : library.mac

; This is the library for the demo programs

;       I/065 entries in eprom

inits   equ     $f00c           Init. a selected I/O device
outx    equ     $f006           Output to selected device

;       DOS65 entries

output  equ     $c023           Print character to screen
input   equ     $c020           Get character from keyboard
prtext  equ     $c03b           Print a textstring until $00

        end
```

De inhoud van prtstr.mac is :

```
; file prtstr.mac

; Author : Erwin Visschedijk


        lib     library

        argl    table1          Default table 1
arg     argm    tabsel          Otherwise take argument table

esc     equ     $1b             Define the escape character

table1  equ     1               Define arguments
table2  equ     2
table3  equ     3

        org     $a000           Utility start address

prtstr  ldx     #2              Select printer to init
        jsr     inits
        bcc     1.f             Branch if successful
        jsr     prtext
        fcc     7,'\rPrinter is not ready.'
        fcc     ' No initialization done.\r',0
        rts                     Error exit
```

```
1       ldy     #0
2       lda     table,y
        ldx     #2              Select printer for output
        jsr     outx
        iny
        cpy     #(tabend-table) Calculate table length
        bne     2.b             Branch if not on end yet
3       rts                     Normal exit

        if      tabsel==table1
table   fcc     esc,'C',0,12    12 inch paper
        fcc     $0f             Condensed mode setting
        fcc     esc,'0'         1/8e inch line spacing
        fcc     esc,'1',10      left marging setting to 10 positions
        fcc     esc,'O'         skip over perforation set cancel
tabend  equ     $               table end
        elseif  tabsel==table2
table   fcc     esc,'C',0,11    11 inch paper
        fcc     $0f             Condensed mode setting
        fcc     esc,'0'         1/8e inch line spacing
        fcc     esc,'1',10      left marging setting to 10 positions
        fcc     esc,'O'         skip over perforation set cancel
tabend  equ     $               table end
        elseif  tabsel==table3
table   fcc     esc,'C',0,12    12 inch paper
        fcc     $0f             Condensed mode setting
        fcc     esc,'0'         1/8e inch line spacing
        fcc     esc,'1',20      left marging setting to 20 positions
        fcc     esc,'O'         skip over perforation set cancel
tabend  equ     $               table end
        endif

        end     prtstr
```

Voorin prtstr.mac zien we de pseudo opcode LIB staan. Hier wordt verwezen naar de library file library.mac. Hierin staan definities die we vaker nodig hebben. Als we diverse programma's schrijven die dezelfde externe routines aanroepen, dan is het erg handig eenmalig een library file aan te maken en deze daarna aan het hoofdprogramma te hangen met LIB.

Verder staat er een argm instructie. Dat houdt in dat er een argument vanuit de aanroep van AS op DOS command niveau kan worden meegegeven. Om dit argument een default waarde te geven wordt daarvoor een argl instructie gegeven met die default waarde erachter. De argumenten die kunnen worden meegegeven moeten wel gedefinieerd worden. Dat gebeurd iets verder in het programma met de equ instructie.

Er moet altijd een org instructie in het programma staan om AS te kennen te geven waar het programma moet komen te staan. Tevens moet er worden afgesloten met een end. Hierna kan commentaar geschreven worden, er wordt dan niet meer geassembleerd.

Er staan drie tabellen gegeven in bovenstaand programma. Afhankelijk van het meegegeven argument wordt een van de tabellen geselecteerd. Dat is duidelijk te zien in de uitvoer.

De commandfile asprt ziet er alsvolgt uit :

```
> -f prt1.lst as -l -bprt1 prtstr
> -f prt2.lst as -l -bprt2 -atable2 prtstr
```

Als deze file met '@ asprt' aangeroepen wordt, worden de volgende files gegenereerd :

- prt1.lst Listfile met output van AS zonder argument
- prt2.lst Listfile met output van AS met argument table2
- prt1.BIN Binary file met object voor 1e programma
- prt2.BIN Binary file met object voor 2e programma

De listfile prt1.lst ziet er alsvolgt uit :

```
                ; file prtstr.mac
                        lib     library
                ; file  : library.mac

                ; This is the library for the demo programs

                ;       I/O65 entries in eprom

          F00C  inits   equ     $f00c           Init. a selected I/O device
          F006  outx    equ     $f006           Output to selected device

                ;       DOS65 entries

          C023  output  equ     $c023           Print character to screen
          C020  input   equ     $c020           Get character from kuyboard
          C03B  prtext  equ     $c03b           Print a textstring until $00

                        end

                        argl    table1          Default table 1
                arg     argm    tabsel          Otherwise take argument table

          001B  esc     equ     $1b             Define the escape character

          0001  table1  equ     1
          0002  table2  equ     2
          0003  table3  equ     3

          A000          org     $a000           Utility start address

A000 A2 02      prtstr  ldx     #2              Select printer to init
A002 20 0CF0            jsr     inits
A005 90 35              bcc     1.f             Branch if successful
A007 20 3BC0            jsr     prtext
A00A 070D507269         fcc     7,'\rPrinter is not ready.'
A00F 6E74657220
A014 6973206E6F
A019 7420726561
A01E 64792E
A021 204E6F2069         fcc     ' No initialization done.\r',0
A026 6E69746961
A02B 6C697A6174
A030 696F6E2064
A035 6F6E652E0D
A03A 00
A03B 60                 rts                     Error exit
```

```
A03C A0 00      1       ldy     #0
A03E B9 4CA0    2       lda     table,y
A041 A2 02              ldx     #2              Select printer for output
A043 20 06F0            jsr     outx
A046 C8                 iny
A047 C0 0C              cpy     #(tabend-table) Calculate table length
A049 D0 F3              bne     2.b             Branch if not on end yet
A04B 60         3       rts                     Normal exit

                        if      table1==table1
A04C 1B43000C   table   fcc     esc,'C',0,12    12 inch paper
A050 0F                 fcc     $0f             Condensed mode setting
A051 1B30               fcc     esc,'0'         1/8e inch line spacing
A053 1B6C0A             fcc     esc,'l',10      left marging setting to 10 positions
A056 1B4F               fcc     esc,'O'         skip over perforation set cancel
         A058   tabend  equ     *               table end
                        elseif  tabsel==table2
                        elseif  tabsel==table3
                        endif

         A000           end     prtstr
```

Errors detected: 0

We zien dat wanneer er geen argument vanuit de aanroep wordt meegegeven dat het default argument genomen wordt.

De listfile prt2.lst ziet er alsvolgt uit :

```
                ; file prtstr.mac

                ; Author : Erwin Visschedijk

                        lib     library
                ; file  : library.mac

                ; This is the library for the demo programs

                ;       I/065 entries in eprom

         F00C   inits   equ     $f00c           Init. a selected I/O device
         F006   outx    equ     $f006           Output to selected device

                ;       DOS65 entries

         C023   output  equ     $c023           Print character to screen
         C020   input   equ     $c020           Get character from keyboard
         C03B   prtext  equ     $c03b           Print a textstring until $00

                        end

                        argl    table1          Default table 1
                arg     argm    tabsel          Otherwise take argument table

         001B   esc     equ     $1b             Define the escape character

         0001   table1  equ     1
         0002   table2  equ     2
         0003   table3  equ     3
```

```
                A000            org     $a000           Utility start address

A000 A2 02      prtstr          ldx     #2              Select printer to init
A002 20 0CF0                    jsr     inits
A005 90 35                      bcc     1.f             Branch if successful
A007 20 3BC0                    jsr     prtext
A00A 070D507269                 fcc     7,'\rPrinter is not ready.'
A00F 6E74657220
A014 6973206E6F
A019 7420726561
A01E 64792E
A021 204E6F2069                 fcc     ' No initialization done.\r',0
A026 6E69746961
A02B 6C697A6174
A030 696F6E2064
A035 6F6E652E0D
A03A 00
A03B 60                         rts                     Error exit

A03C A0 00      1               ldy     #0
A03E B9 4CA0    2               lda     table,y
A041 A2 02                      ldx     #2              Select printer for output
A043 20 06F0                    jsr     outx
A046 C8                         iny
A047 C0 0C                      cpy     #(tabend-table) Calculate table length
A049 D0 F3                      bne     2.b             Branch if not on end yet
A04B 60         3               rts                     Normal exit

                                if      table2==table1
                                elseif  table2==table2
A04C 1B43000B   table           fcc     esc,'C',0,11    11 inch paper
A050 0F                         fcc     $0f             Condensed mode setting
A051 1B30                       fcc     esc,'0'         1/8e inch line spacing
A053 1B6C0A                     fcc     esc,'l',10      left marging setting to 10 positions
A056 1B4F                       fcc     esc,'O'         skip over perforation set cancel
          A058  tabend          equ     *               table end
                                elseif  tabsel==table3
                                endif

          A000                  end     prtstr

Errors detected: 0
```

Nu is het argument table2 genomen. We zien dat de tweede tabel in de object is opgenomen.

De beide .BIN files moeten nu nog omgewerkt worden naar commando's die direct vanuit DOS kunnen worden aangeroepen. Ten eerste moet de file prt1.BIN een andere naam gegeven worden, namelijk de naam die het commando moet hebben. Stel het commando moet heten 'COMA'. Dan geven we het volgende commando 'rename prt1.bin COMA'. Nu moet COMA nog als commando gedefinieerd worden : 'SETMODE -C COMA'. Copieer het commando COMA naar de systeemschijf en we hebben er een commando bij.

Wat veel gedaan wordt is het volgende. De sourcefile heet bv. TYPE.MAC. Na assembleren ontstaat een file TYPE.BIN. Hier moet een commando van gemaakt worden. dat gebeurd door in te typen :
'makecom TYPE'
makecom is een commandfile die op de systeemschijf staat met de volgende inhoud :

```
;
; file makecom
;
; make from .bin file a command, delete .bin file
;
copy U:&1.bin U:&1
setmode -c U:&1
delete -y U:&1.bin
;
```

Wat hier gebeurd is het volgende :
Eerst wordt de file TYPE.BIN gecopieerd naar TYPE. Dat alles in de user directory (U:). Daarna wordt de mode van de file TYPE omgezet tot command met SETMODE -C. Tot slot wordt de file TYPE.BIN gedelete om weer ruimte op de schijf te scheppen daar we die file toch niet meer nodig hebben. Het commando TYPE kan worden uitgeprobeerd in de user directory met U:TYPE <cr>. Eventueel kan het commando naar de systeemschijf gecopieerd worden.

Hieronder staat een programma dat u zelf eens kunt uitproberen door het te assembleren en er een commando van te maken.

```
; file star.mac

; usage as demo

; Author E.J.M. Visschedijk

        org     $a000

output  equ     $c023
crlf    equ     $c02f

star    ldx     #10
        lda     #'*'
1       jsr     output
        dex
        bne     1.b
        jmp     crlf

        end     star            Don't forget the label name star!!!
```

Noem bovenstaand programma star.mac.
1- Type in 'as star' om het programma te assembleren.
2- Type in 'copy star.bin star'
3- Type in 'setmode -c star' om er een commando van te maken.
4- Type in 'delete star.bin' om overtollige files weg te gooien.
Als nu de commandfile 'makecom' al op uw systeemschijf had gestaan, dan had u ook vanaf punt 2 in kunnen typen : 'makecom star' om hetzelfde te bereiken.
5- Type nu in 'u:star' om het programma uit te proberen.

Om te zien waar het programma in het geheugen wordt geladen en wat het startadres is kan het commando MAP van dienst zijn. We typen dan 'map star' het resultaat is:
A000-A00C
A000
Het programma loopt van $A000 tot $A00C en het startadres is $A000. Als er geen startadres komt, dan is de labelnaam 'star' achter het 'end' statement op de laatste regel in de file 'star.mac' vergeten en kan het programma nooit als commando fungeren.

Om de code van het programma 'star' op adres $8000 in het geheugen te laden wordt ingetypt:
OLOAD star E000

Er volgt nu een voorbeeld voor het gebruik van macro's. De IO65 routine POSIT zet de cursor op de positie X,Y op het beeldscherm. Voordat we de routine aanroepen moet eerst het X en het Y register geladen worden met de positie die we in gedachten hebben. Hoe handig zou het niet zijn als we in onze programmatuur echter alleen maar hoeven in te typen:
'pos 7,3' om de cursor op positie 7 in de X en 3 in de Y richting te plaatsen. We definieren daarvoor de macro pos. We bekijken het onderstaande programma:

```
; file macro.mac

; Example for macro usage

posit   equ     $f024
output  equ     $c023

; Macro definition

pos     macro   xpos,ypos
        ldx     #xpos
        ldy     #ypos
        jsr     posit
        endm

        org     $a000

prog    lda     #$0c
        jsr     output          Clear screen
        pos     20,4            Position the cursor
        lda     #'*'
        jsr     output          Print *
        pos     40,10           Position the cursor
        lda     #'+'
        jsr     output          Print +
        pos     1,12            Position the cursor
        rts

        end     prog
```

We zien dat de regels waarin pos aangeroepen wordt geexpandeerd worden. Automatisch wordt er voor de macro aanroep met zijn argumenten de complete macro ingevuld.

```
                ; file macro.mac

                ; Example for macro usage

          F024  posit   equ     $f024
          C023  output  equ     $c023

                ; Macro definition

                pos     macro   xpos,ypos
                        ldx     #xpos
                        ldy     #ypos
                        jsr     posit
                        endm

          A000          org     $a000

A000 A9 0C      prog    lda     #$0c
A002 20 23C0            jsr     output          Clear screen
                        pos     20,4            Position the cursor
A005 A2 14              ldx     #20
A007 A0 04              ldy     #4
A009 20 24F0            jsr     posit
A00C A9 2A              lda     #'*'
A00E 20 23C0            jsr     output          Print *
                        pos     40,10           Position the cursor
A011 A2 28              ldx     #40
A013 A0 0A              ldy     #10
A015 20 24F0            jsr     posit
A018 A9 2B              lda     #'+'
A01A 20 23C0            jsr     output          Print +
                        pos     1,12            Position the cursor
A01D A2 01              ldx     #1
A01F A0 0C              ldy     #12
A021 20 24F0            jsr     posit
A024 60                 rts

          A000          end     prog
```

Errors detected: 0

Eventueel kan met de optie nomac onderdrukt worden dat in de listing de expansie zichtbaar wordt. Dan wordt ergens voorin het programma gezet :

```
        opt     nomac
```