

BUT ET DOMAINE D'APPLICATION

CALM fixe les notations pour des programmes en langage d'assemblage, qui sont utilisables pour des processeurs d'une taille des mots de données et d'adresses quelconque et qui sont basés sur une architecture de "von-Neumann". Ces notations décrivent à l'aide de verbes anglais l'opération à effectuer. Mais aussi des architectures plus complexes et futures sont prévues.

CALM fixe les noms mnémotechniques et la syntaxe des opérations, conditions de test, registres spéciaux, types de données, modes d'adressage et pseudo-instructions les plus utilisés.

La notation CALM décrit précisément tous les instructions d'un processeur. S'il y a pour une opération plusieurs (mais fonctionnellement identiques) instructions au choix (p.ex. adressage absolu et relatif), CALM offre d'une part une notation explicite et précise pour chacune de ces instructions et d'autre part une notation simplifiée, qui laisse à l'assembleur le choix de l'instruction.

CALM laisse une très grande liberté de mise en page, mais permet et encourage une présentation claire et modulaire des programmes. CALM tient aussi en compte des affichages et imprimantes futurs. Des modules et les pseudo-instructions associées sont aussi prévues et permettent ainsi le lien avec des langages de programmation évolués.

CALM ne définit pas de format binaire pour le programme objet, mais recommande la norme P695. CALM a été normalisé dans DIN 66283.

TERMES

Les termes utilisés correspondent aux significations comme elles sont utilisées par les fabricants de microprocesseurs.

Les diagrammes de syntaxe font foi et correspondent aux règles usuelles comme ils sont utilisés par exemple dans le langage Pascal. Des cercles et des rectangles avec des demi-cercles se réfèrent à des éléments de base (caractère ou mots-clés). Des rectangles se réfèrent à des éléments définis ailleurs. Un symbole entre parenthèses pointues (<>) indique la valeur de ce symbole, p.ex. <CR>.

Les nombres décimaux sont appelés ici simplement nombres. Ces nombres restent aussi comme indicateurs supplémentaires décimaux, même si la base par défaut a été changée.

JEU DE CARACTERES

Le jeu de caractères est basé sur le jeu de caractères ASCII complét. Il se compose de caractères imprimables et non imprimables.

CARACTERES IMPRIMABLES

* Lettres

- obligatoire: A..Z

- facultatif: a..z

Les lettres minuscules et majuscules dans les symboles et les noms des registres réservés sont traitées identiques.

* Chiffres

- obligatoire: 0..9

* Signes:

- obligatoire:

+	signe plus	-	signe moins
*	astérisque	/	barre oblique
.	point	,	virgule
:	deux-points	=	signe égal
;	point-virgule	_	souligné
\$	signe dollar	#	dièze
'	apostrophe	"	guillemets
^	accent circonflexe		barre verticale
%	signe pourcent	()	parenthèses
{}	accolades	<>	parenthèses pointues
<ESPACE>	espace	<TAB>	tabulateur horizontale

SYMBOLES DEFINIS PAR L'UTILISATEUR

Un symbole défini par l'utilisateur est introduit explicitement par le programmeur. Ce symbole désigne une constante ou une adresse mémoire et possède des attributs supplémentaires cachés, qui aident à l'assemblage et permettent la détection d'erreurs. On ne peut pas utiliser des symboles réservés ou prédéfinis.

Il y a quatre types de symboles définis par l'utilisateur:

- étiquette globale
- étiquette locale
- assignation
- pseudo-instruction IMPORT

ETIQUETTE GLOBALE

Une étiquette globale est un symbole qui est défini dans le champ d'étiquette d'un ordre d'assembleur. La valeur de l'étiquette est donnée par la valeur du compteur d'adresses de l'assembleur. Une étiquette globale est valable dans le programme entier.

Le diagramme de syntaxe est illustré à la figure 9.

étiquette globale

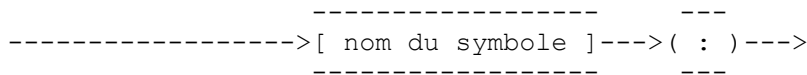


Fig. 9

ETIQUETTE LOCALE

Une étiquette locale existe seulement entre deux étiquettes globales. La longueur du nom du symbole est limitée à 8 caractères. Le diagramme de syntaxe est donné à la figure 10.

étiquette locale

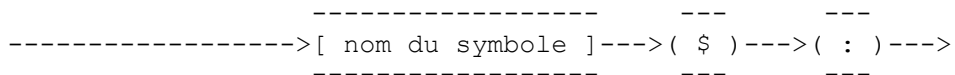
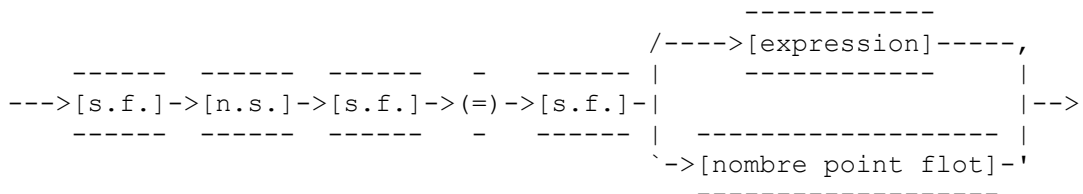


Fig. 10

ASSIGNATION

Un symbole est défini par le signe égal (=). La partie droite de l'assignation est une expression ou un nombre point flottant. Le diagramme de syntaxe est donné à la figure 11.

assignation



n.s.: nom du symbole

Fig. 11

LA PSEUDO-INSTRUCTION IMPORT

Une pseudo-instruction IMPORT déclare les symboles définis dans d'autres modules de programme. Le nom réservé .IMPORT est suivi par une liste de symboles. Le diagramme de syntaxe est donné à la fig. 12. Les valeurs des symboles sont inconnues au moment de l'assemblage.

pseudo-instruction IMPORT

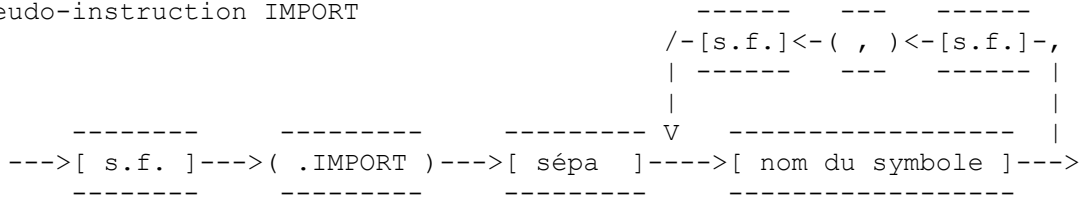


Fig. 12

EXPRESSIONS

Les expressions avec les opérateurs de plus haute priorité sont évaluées en premier. L'évaluation se fait de gauche à droite pour des opérateurs de priorité égale. On peut changer la priorité avec des parenthèses.

Le diagramme de syntaxe est donné à la fig. 13. Les opérateurs sous facteur ont la plus haute priorité, les opérateurs de comparaison la plus faible.

Opérateurs avec un opérande (opérateurs monadiques):

- + positif
- négatif (complément à deux)
- .BIT. équivalent à 1.sl.facteur ou 2**facteur
- .DEFINED. retourne TRUE si symbole est défini sinon FALSE
- .EQ0. négation logique (0 -> TRUE, <>0 -> FALSE)
- .HIGH8. équivalent à (facteur.sr.8).and.16'FF
- .HIGH16. équivalent à (facteur.sr.16).and.16'FFFF
- .LOG2. retourne 0..31 si facteur = 1..16'FFFFFFFF (0 = erreur)
- .LOW8. équivalent à facteur.and.16'FF
- .LOW16. équivalent à facteur.and.16'FFFF
- .NOT. complément (complément à un)
- .SQR. équivalent à facteur*facteur
- .SQRT. retourne la racine carrée (facteur = 0..16'7FFFFFFFF)

Opérateurs avec deux opérandes (opérateurs diadiques):

- + addition
- soustraction
- * multiplication
- ** élever à une puissance
- / division (ou .DIV.)
- .MOD. reste de la division entier
- .OR. ou logique, ou inclusif
- .AND. et logique
- .XOR. ou exclusif
- .ASR. décalage arithmétique (2e opérande: amplitude)
- .SR./.RR. décalage/rotation à droite
- .SL./.RL. décalage/rotation à gauche (.ASL. = .SL.)

Opérateurs de comparaison (2 opérandes, résultat est TRUE ou FALSE)

- .EQ. égal
- .NE. différent
- .LO. inférieur * non-signé,
- .LS. inférieur ou égal * logique
- .HS. supérieur ou égal *
- .HI. supérieur *
- .LT. plus petit que + arithmétique,
- .LE. plus petit que ou égal + complément à deux
- .GE. plus grand que ou égal +
- .GT. plus grand que +

indicateur d'adresses

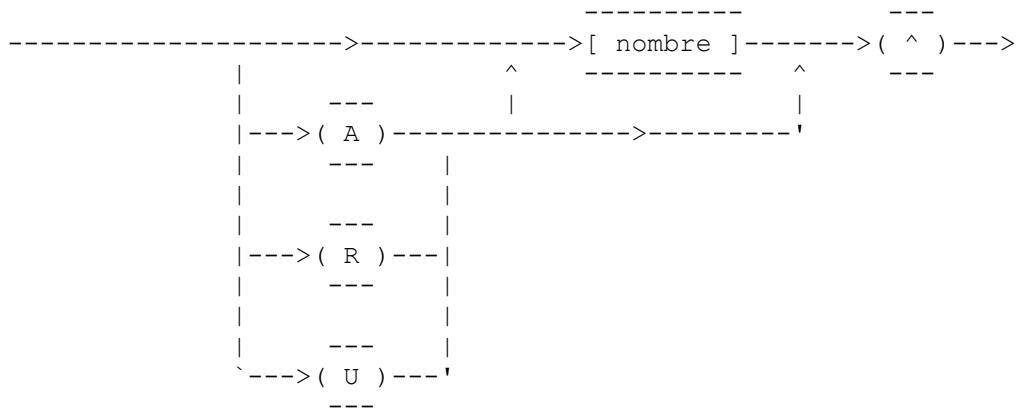


Fig. 15

ADRESSES, ESPACES D'ADRESSAGE ET ADRESSAGE DIRECT

Un espace d'adressage est une suite de locations mémoire qui sont numérotées à partir de zéro. Chaque location est souvent 8 bits (nommé un octet) de large et est déterminé par un numéro unique ou une expression équivalente appelé son adresse. S'il y a plusieurs espaces d'adressage, la définition précédente se réfère à l'espace d'adressage où se trouvent les instructions du programme. Cet espace d'adressage est appelé espace mémoire. Les autres espaces d'adressage (entrée/sortie, données) sont indiqués par des caractères spéciaux réservés (\$, %).

Les espaces d'adressage des registres ne sont souvent pas ordonnés. Les noms des registres sont réservés. Des registres numérotés peuvent être considérés comme des espaces d'adressage qui commencent avec les lettres A, D ou R.

L'adressage d'un espace d'adressage par son adresse (nom d'un registre ou expression) est appelé adressage direct. Le diagramme de syntaxe est donné à la fig. 16.

adressage direct

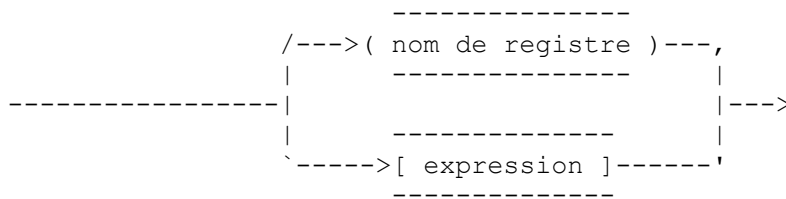


Fig. 16

ADRESSAGE INDIRECT

Le contenu d'une adresse quelconque dans un espace d'adressage quelconque peut de nouveau être utilisé comme une adresse dans un espace d'adressage quelconque. Ceci est appelé adressage indirect.

Si cet espace n'est pas l'espace mémoire principale, alors il faut indiquer l'espace d'adressage considéré.

Des accolades sont utilisées à indiquer que l'opérande est réutilisé pour calculer la nouvelle adresse pendant l'exécution de l'instruction.

Le diagramme de syntaxe d'une adresse indirecte est illustré à la fig. 17. Le terme adresse effective est défini plus tard.

On ne peut pas construire une adresse de registre par l'adressage indirecte. Si ceci est possible sur un processeur particulier, il faut considérer les registres comme espace mémoire ou espace de données supplémentaire.

ADRESSAGE IMMEDIAT

Dans l'adressage immédiat on utilise l'adresse effective elle-même comme opérande. Ce mode d'adressage, réservé pour l'opérande source, est spécifié avec le dièse comme premier élément.

La même notation est utilisée pour les opérandes avec des valeurs arithmétiques (entier ou point flottant).

Le diagramme de syntaxe d'une adresse immédiate est donné à la fig. 20.

Si l'instruction limite la taille de la valeur immédiate (p.ex. instruction ADD-QUICK), on peut spécifier sa taille avec un indicateur d'adresses ou de données.

adresse immédiate

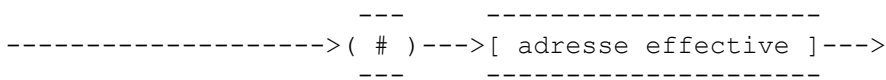


Fig. 20

ADRESSAGE DE BITS

L'adresse d'octet définie précédemment pointe au bit 0 de l'octet, c.à.d. le bit de poids le plus faible (LSB) de cet octet. Les bits sont numérotés de 0 à 7 dans l'octet adressé, de 8 à 15 dans l'octet suivant, etc. Des adresses de bits négatives sont possibles: les bits dans l'octet précédant sont numérotés de -8 à -1, où le bit -8 est le LSB.

L'adresse de bit est une extension de l'adresse d'octet. Tous les modes d'adressage sont permis. Un deux-points (:) sépare l'adresse d'octet de l'adresse de bit. Le diagramme de syntaxe est donné à la fig. 21.

adresse de bit

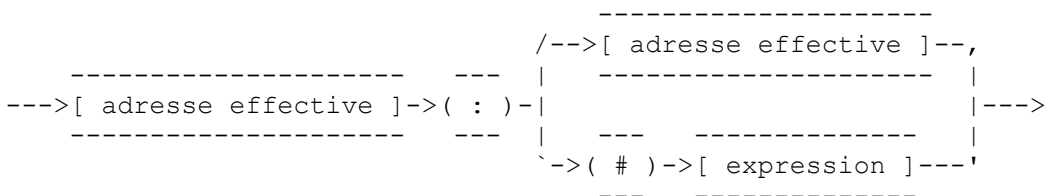


Fig. 21

LISTES

Une liste est une énumération de registres, locations mémoire ou bits. Des registres, locations mémoire ou bits consécutifs peuvent être réunis dans un groupe.

Une liste de registres ou locations mémoire est soit défini par des groupes (le premier et le dernier élément, le premier élément et la longueur), soit par des éléments individuels. Une barre verticale sépare les énumérations individuels.

Le diagramme de syntaxe d'une liste de registres ou de locations mémoire est illustré à la fig. 22.

CODES CONDITIONS

L'exécution d'une instruction peut dépendre de l'état d'un indicateur. Un indicateur lui-même peut être modifié par une instruction. Un code condition exprime l'état d'un indicateur ou la combinaison d'états d'indicateurs. Les codes conditions sont des noms mnémotechniques qui sont composés usuellement de deux lettres:

EQ	égal	
NE	différent	
BC	bit à zéro (EQ)	
BS	bit à un (NE)	
CS	report à un	
CC	report à zéro	
VS	dépassement à un	
VC	dépassement à zéro	
MI	moins, négatif	
PL	plus, positif	
LO	inférieur	après comparaison non signée
LS	inférieur ou égal	
HS	supérieur ou égal	
HI	supérieur	
LT	plus petit que	après comparaison arithmétique
LE	plus petit que ou égal	
GE	plus grand que ou égal	
GT	plus grand que	

Les instructions souvent combinées: JUMP, DJ, CALL, RET und SKIP.