CALM
Common **A**ssembly **L**anguage for **M**icroprocessors

Définitions

Patrick Fäh, La Colombière, CH-1783 Pensier, Suisse

Table des matières

BUT ET DOMAINE D'APPLICATION

CALM fixe les notations pour des programmes en langage d'assemblage, qui sont utilisables pour des 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 fonctionellement 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). Ex: .fop.

Des rectangles se réfèrent à des éléments définis ailleurs. Ex: commentaire

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	_	ouligné
$	signe dollar	#	dièze
'	apostrophe	"	uillemets
^	accent circonflexe	|	barre verticale
%	igne pourcent	)	parenthèses
{}	accolades	<>	parenthèses pointues
<ESPACE>	espace	<TAB>	tabulateur horizontale

- utilisable facultativement dans des commentaires:

&	signe "et"	!	point d'exclamation
?	point d'interrogation	@	Signe "at"
`	accent grave	~	tilde
[]	crochets	\	barre oblique en arrière

Remarque: La barre oblique en arrière ou caractère d'échappement typographique permet l'extension du jeu de caractères par des caractères spéciaux, graphiques et typographiques. La compatibilité entre un écran ou une imprimante à chasse fixe et d'une unité avec chasse variable et des tabulateurs librement positionnables peut être atteinte par des pseudo-tabulateurs (\<TAB>).

Caractères non ASCII

Des caractères avec accents ou d'autres signes spéciaux sont utilisables dans des commentaires s'il n'y a pas de contradiction avec les caractères définis ci-dessus.

CARACTERES NON IMPRIMABLES

caractères de commande

<CR> fin de ligne

<NULL> vide

Tous les autres caractères de commande sont ignorés.

PROGRAMME EN ASSEMBLEUR ET ORDRES

Un programme en langage d'assemblage est une suite finie d'ordres. Chaque ordre peut avoir jusqu'à 255 caractères et est terminé par un caractère de retour de chariot <CR>. L'ordre END termine un programme en langage d'assemblage. Le diagramme de syntaxe d'un programme en langage d'assemblage est donné à la fig. 1.

Le diagramme de syntaxe d'un ordre est illustré à la fig. 2.

programme en langage d'assemblage

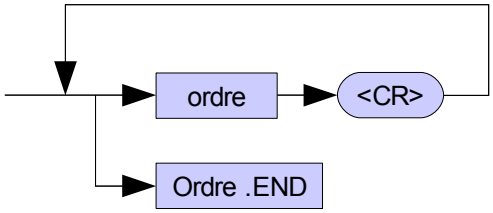
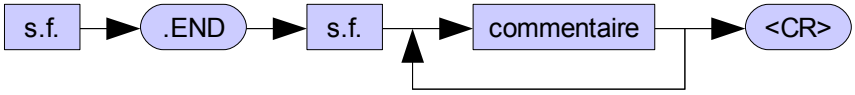


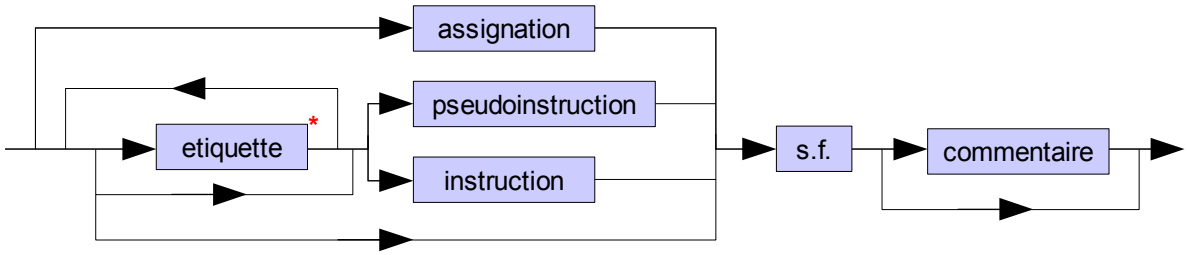
Figure 1



s.f. : Séparateur facultatif

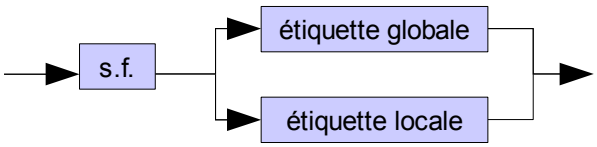
Figure 2

ordre

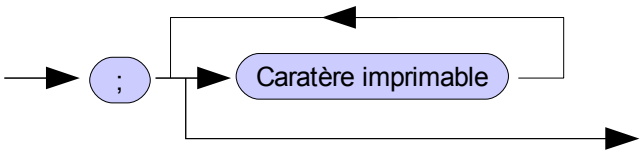


* pour quelques pseudo-instructions des étiquettes ne sont pas admises.

étiquette



commentaire



SEPARATEURS

Un séparateur est une suite non vide d'espaces ou / et de tabulateurs. Dans la pratique, on utilise qu'un espace ou qu'un tabulateur. Le diagramme de syntaxe est présenté à la fig. 3a. Le diagramme de syntaxe d'un séparateur facultatif (s.f.) est donné à la fig. 3b.

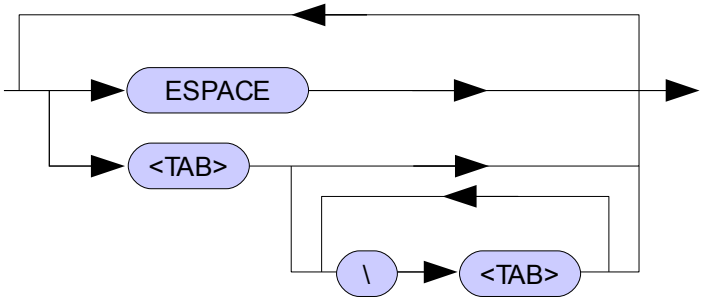


Fig. 3A Séparateur

Fig. 3B s.f.

CONSTANTES NON SIGNEES

Des constantes non signées sont des p-nombres ou des codes caractères. Elles sont mémorisées à l'intérieur de l'assembleur avec une résolution de 128 bits. La fig. 4 montre le diagramme de syntaxe.

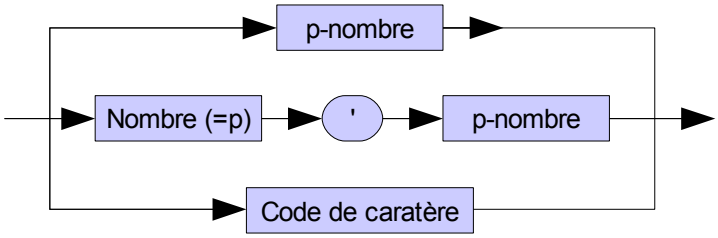


Fig. 4 constante non signée

P-NOMBRES

Un p-nombre est un nombre entier positif de la base p avec 2 <= p <= 16. Les p-chiffres correspondants sont pris de la liste ordonnée 0..9, A..F. Le diagramme de syntaxe est donné à la fig. 5.

Si la base par défaut n'est pas p, on peut précéder le nombre explicitement par la base:

p=2	(binaire)	2'	p.ex.:	2'11110001
p=8	(octal)	8'	p.ex.:	8'361
p=10	(décimal)	10'	p.ex.:	10'241
p=16	(sexadécimal)	16'	p.ex.:	16'F1

La base par défaut est initialisée comme étant la base 10 dans l'assembleur CALM. Si la base est supérieure à 10 et aucun préfixe est utilisé, alors le premier chiffre d'un nombre doit être un chiffre décimal. Ainsi un zéro non significatif doit être placé devant le p-nombre, si le premier p-chiffre est une lettre (p.ex. 0FF).

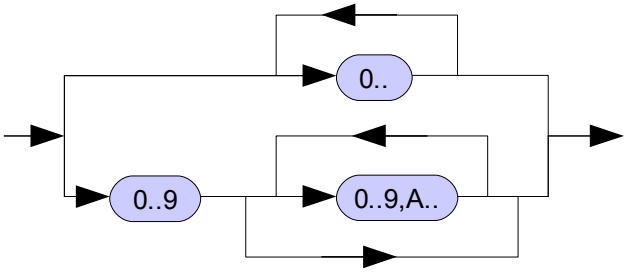


Fig. 5 p-nombre

CODE CARACTERE

Un caractère imprimable entre guillemets fournit le code ASCII de e caractère. Le diagramme de syntaxe est donné à la fig. 6.

Exemple: "A" fournit 16'41 et avec """" on obtient le code ASCII 16'22 des guillemets.

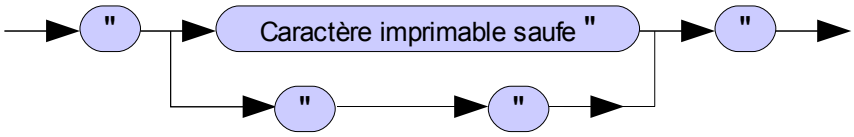


Fig. 6 code caractère

NOMBRE POINT FLOTTANT

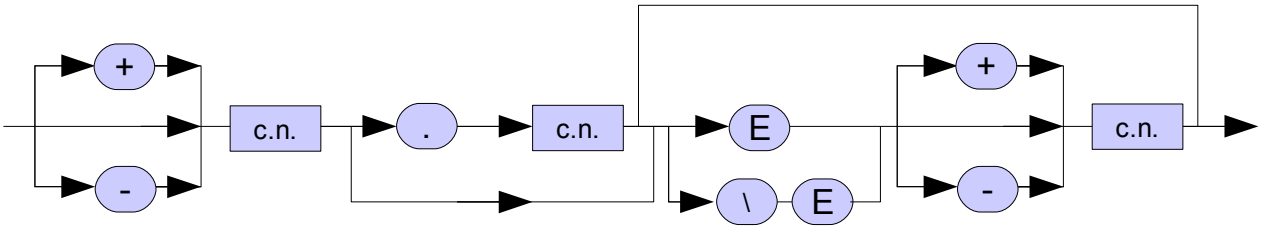
Le diagramme de syntaxe de nombres point flottant est donné à la figure 7. On suppose le format IEEE P754.

L'assembleur utilise internement le format P754 80 bits. La valeur d'un nombre point flottant sera convertie dans le format de 64 ou 32 bits quand la valeur est utilisée dans une instruction.

Les nombres point flottant sont seulement utilisables comme valeurs immédiates. Le traitement de nombres point flottant dans des expressions n'est pas demandé en CALM.

Exemple: -300.0 ou -3E+2.

Remarque: Nous utilisons ici le terme nombre point flottant (au lieu nombre virgule flottant), car il faut bien utiliser un point.



c.n.: constante non-signée

Fig. 7 nombre point flottant

SYMBOLES

Symboles ont un nom et une valeur. Le nom du symbole peut contenir jusqu'à 32 caractères. L'assembleur mémorise la valeur du symbole avec une longueur de mot de 256 bits. Les lettres minuscules sont équivalentes aux lettres majuscules correspondantes.

Le diagramme de syntaxe d'un nom de symbole est donné à la fig. 8.

Il y a trois types de symboles:

- symboles réservés
- symboles prédéfinis
- symboles définis par l'utilisateur

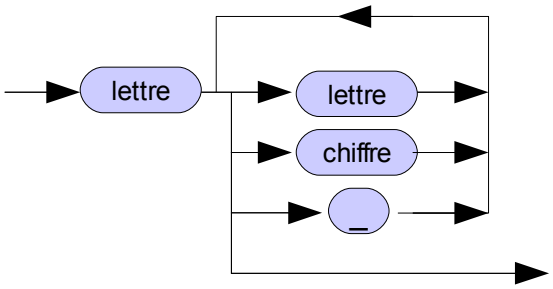


Fig. 8 nom du symbole

SYMBOLES RESERVES

Les symboles réservés désignent des registres spéciaux. S'il ne faut pas respecter une architecture particulière, les noms suivants sont réservés:

PC le compteur d'adresses au moment de l'exécution du programme (Program Counter)
SP le pointeur de pile (Stack Pointer)
F le registre d'indicateurs arithmétique (Flags)
S le registre d'état (System, Status)

Les noms des registres doivent correspondre à ceux du fabricant. Tableau 1 montre les noms des registres choisis en CALM pour quelques processeurs.

Z80:								
Zilog:	F	A	BC	DE	HL	IX	IY	SP
CALM:	F	A	BC	DE	HL	IX	IY	SP
6809:								
Motorola:	CC	D	X	Y	S	U	DP	
CALM:	F	AB	IX	IY	SS	US	P	
68000:								
Motorola:	CCR	SR	D0..D7	A0..A7	A7			
CALM:	F	SF	D0..D7	A0..A7	A7 = SP			

Tab. 1

SYMBOLES PREDEFINIS

Il y a trois types de symboles prédéfinis:

-obligatoire:

APC = la valeur du compteur d'adresses de l'asembleur au début de l'instruction dans laquelle APC apparaît
TRUE = -1 (tous les bits à 1)
FALSE = 0 (tous les bits à 0)

-facultatif:

SYSTIME = 0 h m s (32 bits BCD)
SYSDATE = 0 a m j (32 bits BCD)
FLOATPI = nombre pi
FLOATE = nombre e

-défini dans l'assembleur: à éviter

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.

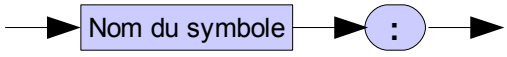


Fig. 9 étiquette globale

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.

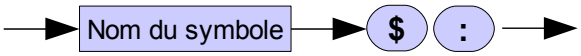


Fig. 10 étiquette locale

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.

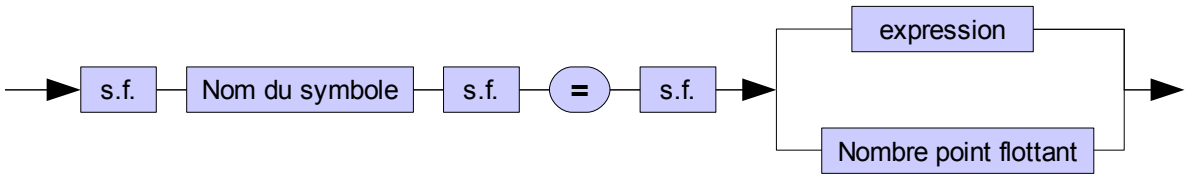


Fig. 11 assignation

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.

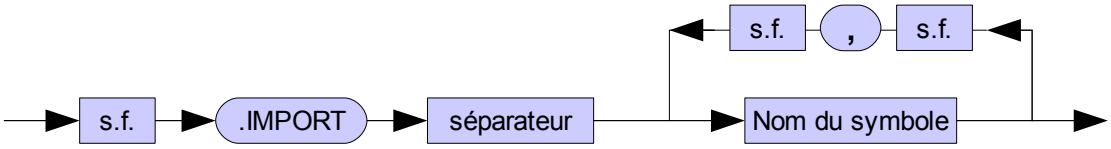


Fig. 12 pseudo-instruction IMPORT

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):

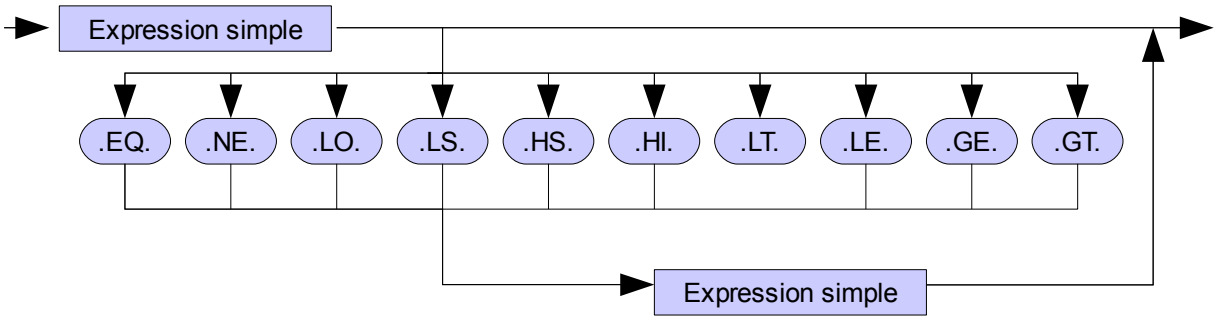
+	positif
-	negatif (complément à deux)
.BIT.	équivalent à 1.sl.facteur ou 2**facteur
.DEFINED.	retourne TRUE si symbole est défini sinon FALSE
.EQ0.	négation logique (0 -> TRUE, <>0 -> FALSE)
.HIGH8.	équivalent à (facteur.sr.8).and.16'FF
.HIGH16.	équivalent à (facteur.sr.16).and.16'FFFF
.LOG2.	retourne 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..16'7FFFFFFF)

Opérateurs avec deux opérandes (opérateurs diadiques):

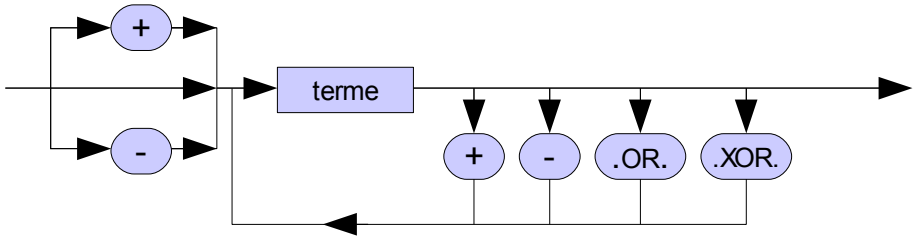
+	addition
-	soustraction
*	multiplication
**	élever à une puissance
/	division (ou .DIV.)
.MOD.	reste de la division entier
.OR.	ou logique, ou inclusif
.AND.	et logique
.XOR.	ou exclusif
.ASR.	décalage arithmétique (2e opérande: amplitude)
.SR./.RR.	décalage/rotation à droite
.SL./.RL.	décalage/rotation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

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

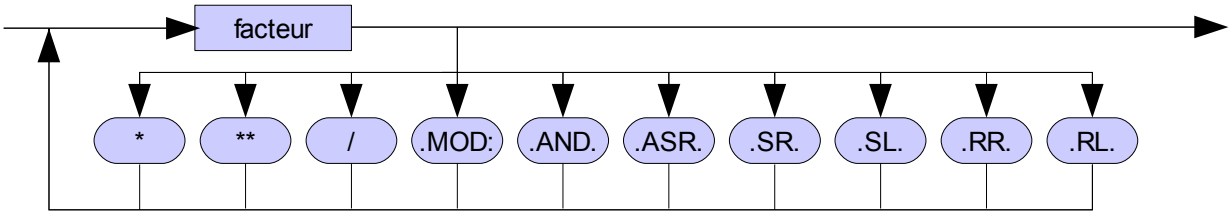
.EQ.	égal	
.NE.	différent	
.LO.	inférieur	* non-signé,
.LS.	inférieur ou égal	* logique
.HS.	supérieur ou égal	*
.HI.	supérieur	*
.LT.	plus petit que	+ arithmétique,
.LE.	plus petit que ou égal	+ complément à deux
.GE.	plus grand que ou égal	+
.GT.	plus grand que	+



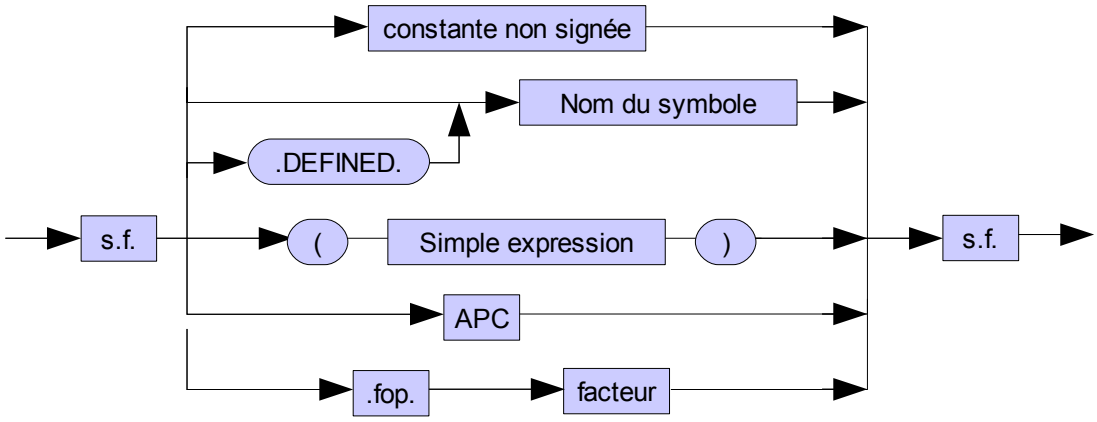
expression



simple expression



terme



facteur

fop = BIT, EQ0, HIGH8, HIGH16, LOG2, LOW8, LOW16, NOT, SQR, SQRT.

Fig. 13

INDICATEUR DE DONNEES

Un indicateur de données est ajouté directement soit au code opératoire soit aux opérandes. Un point est utilisé comme caractère de séparation. Un indicateur de données se compose d'un type de données et d'une taille. Cette taille est toujours exprimée en bits même pour des chiffres BCD et des caractères. S'il n'y a pas de type de données indiqué, il s'agit de données non signées.

Les types de données et leurs lettres réservées associées sont:

U ou rien non signé, logique
A arithmétique (complément à deux)
S avec signe (signe et valeur absolue)
O déplacement (2n-1 biais)
D décimal (non signé, 2 chiffres par octet)
F point flottant (format IEEE 754)

On peut définir d'autres types de données par des lettres supplémentaires pour des processeurs spéciaux. Le diagramme de syntaxe pour l'indicateur de données est donné à la fig. 14.

Dans une instruction, il faut indiquer explicitement les données traitées par des indicateurs de données. Pour des processeurs simples, c.à.d. qui ont pour chaque instruction qu'un type de données et qu'une taille de données, cet indicateur de données peut être supprimé, car il est implicite.

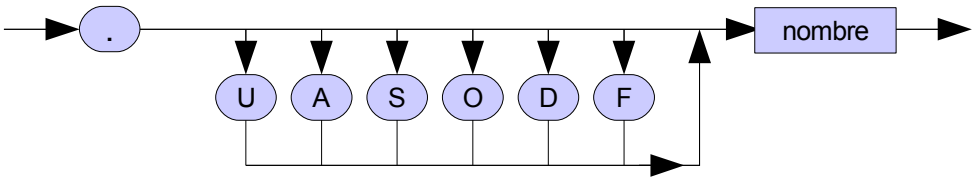


Fig. 14 indicateur de données

INDICATEUR D'ADRESSES

On peut fixer une adresse, représentée par une expression quelconque, dans un mode d'adressage absolu ou relatif par un indicateur d'adresses placé devant cette expression. Un accent circonflexe est utilisé comme caractère de séparation.

Un indicateur d'adresses utilisé pour l'adressage absolu spécifie comment l'adresse donnée doit être étendue à la taille de l'adresse entière. L'extension est soit non signé (extension de zéro, rien ou lettre U) soit arithmétique (extension du signe, lettre A), suivi par le nombre de bits à prendre de l'adresse non étendue.

L'indicateur d'adresses R va être défini sous "adressage relatif" et est utilisé pour l'adressage relatif.

Le diagramme de syntaxe d'un indicateur d'adresses est donné à lafig. 15.

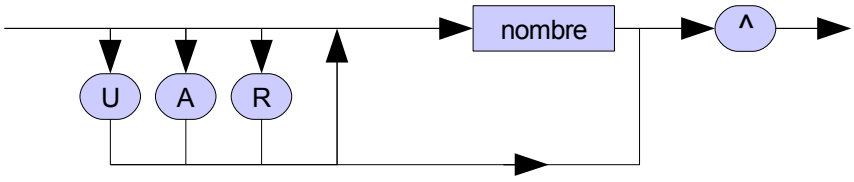


Fig. 15 indicateur d'adresses

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.

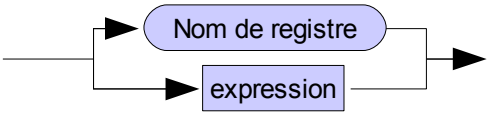


Fig. 16 adressage direct

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.

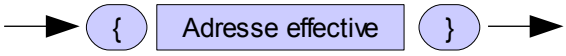


Fig. 17 adresse indirecte

ADRESSAGE RELATIF

Puisque le PC (compteur d'adresses) a une valeur connue au moment de l'assemblage, l'adressage indirect relatif par rapport au PC joue un rôle particulier. Pour le programmeur, l'expression {PC}+DEPLACEMENT est équivalente à une adresse directe ADMEMOIRE (uniquement dans l'espace mémoire). L'assembleur évalue la valeur DEPLACEMENT à partir de la valeur de ADMEMOIRE et place le résultat dans l'instruction.

Le programmeur peut forcer l'adressage relatif en utilisant la lettre R (indicateur d'adresses). Si le déplacement est limité, on ajoute derrière la lettre R le nombre de bits. DEPLACEMENT est toujours codé comme un nombre arithmétique (complément à deux) et est aussi étendu de signe jusqu'à la taille du PC.

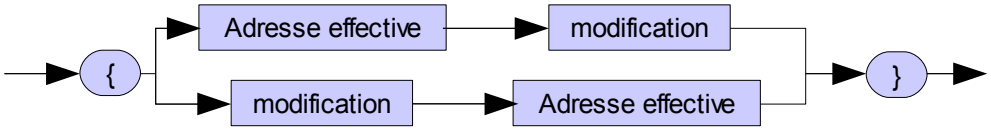
L'indicateur relatif R a été présenté dans la fig. 15.

AUTOMODIFICATION

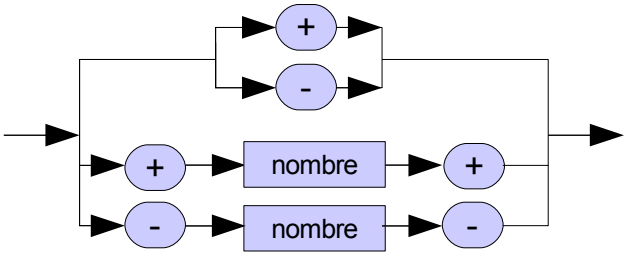
Registres et locations mémoire peuvent être modifiés s'ils sont utilisés pour l'adressage indirect. Incrémentation ou décrémentation par 1, 2 ou 4 est l'opération la plus fréquente.

Un seul signe plus ou moins est utilisé lorsque la valeur incrémenté/ décrémenté dépend de la taille de l'opérande (1, 2, 4 pour 8, 16, 32 bits). Et deux signes plus ou moins encadrant la valeur d'incrémentation/décrémentation si cette valeur ne dépend pas de la taille de l'opérande.

Le diagramme de syntaxe est donné à la fig. 18.



adresse indirecte automodifiée



modification

Fig. 18

ADRESSE EFFECTIVE

Le processeur peut calculer une adresse à partir d'informations provenant directement ou indirectement du champ d'instruction.

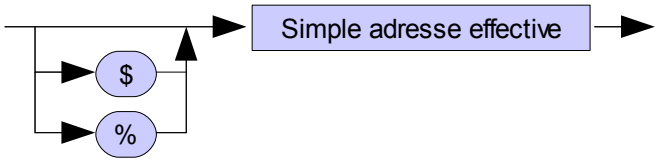
Une adresse effective est la somme d'adresses différentes dans le même espace d'adressage. Chaque adresse peut se trouver dans un sous-espace donné et l'additionneur peut avoir une taille limitée.

La taille de l'additionneur et la manière avec laquelle les adresses sont étendues sont données par des indicateurs d'adresses facultatifs (uniquement nécessaire pour des instructions spéciaux).

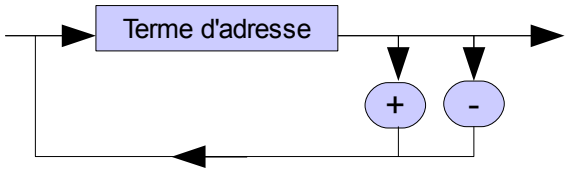
Les opérations entre les adresses sont évaluées pendant l'exécution de l'instruction. Les opérations possibles sont l'addition (+), la soustraction (-) et la multiplication (*).

L'opérateur * est utilisé pour le scaling automatique, c.à.d. La multiplication du facteur d'adresse par 2, 4, etc. dépendant de la taille des éléments dans un tableau. Des parenthèses sont utilisées pour regrouper des expressions. Le diagramme de syntaxe récursif d'une adresse effective est donné à la fig. 19.

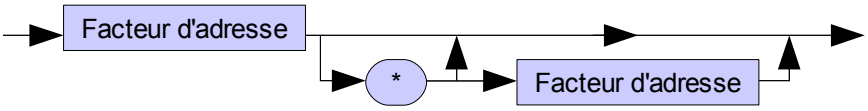
S'il n'y a pas d'indicateurs d'adresses utilisés, l'assembleur choisit l'instruction qui est la plus compacte et la plus rapide.



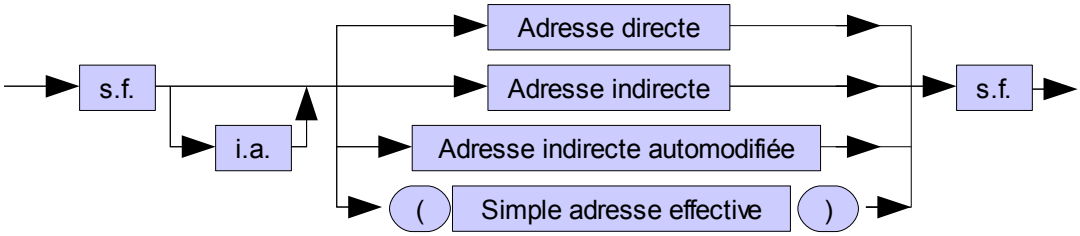
adresse effective



simple adresse effective



terme d'adresse



i.a.: indicateur d'adresses facteur d'adresse

Fig. 19

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 dièze 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.

Fig. 20 adresse immédiate

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.

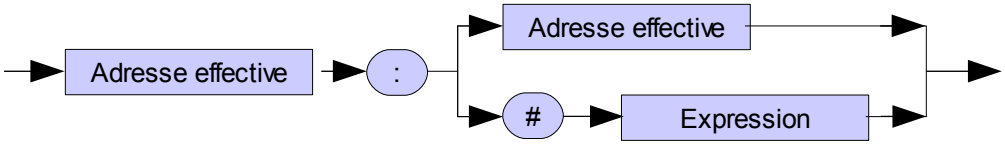


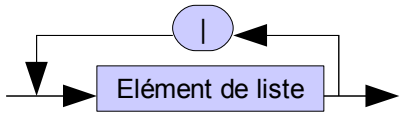
Fig. 21 adresse de bit

LISTES

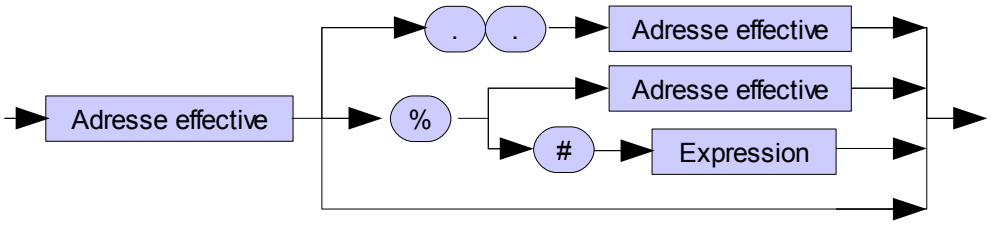
Une liste est une énumération de registres, locations mémoire ou bits. Des registres, locations mémoire ou bits consécutifs peuvent être réunis dans un groupe.

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.



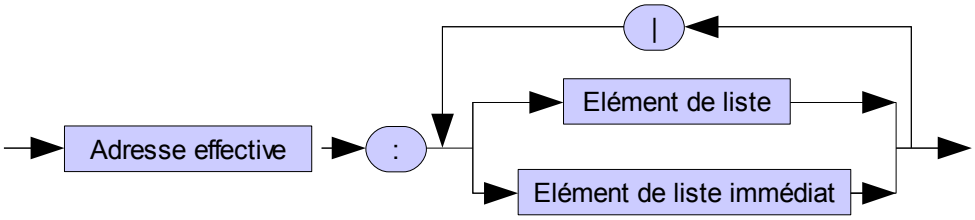
liste de registres ou de locations mémoire



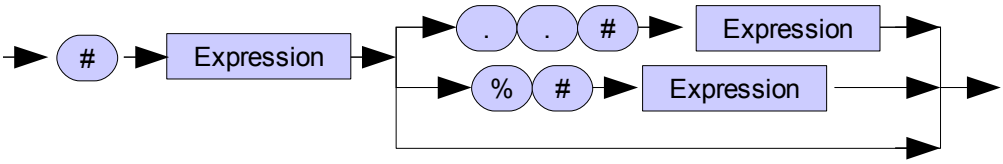
élément de liste

Fig. 22

Une liste de bits est définie ou bien par des groupes (la première et la dernière adresse de bit, la première adresse de bit et la longueur en bits) ou bien par chaque adresse de bit individuelle. Une barre verticale sépare les bits individuels et les groupes de bits. Le diagramme de syntaxe d'une liste de bits est donné à la figure 23.



liste de bits



élément de liste immédiat

Fig. 23

BITS DES REGISTRES D'INDICATEURS ET D'ETAT

Les bits dans les registres d'indicateurs et d'état (F et S) des processeurs sont souvent indiqués par des lettres. Ces lettres ne sont pas des adresses, mais des lettres mnémotechniques ajoutées aux codes opératoires.

Les lettres recommendées sont données ci-dessous. Il faut éviter à tout prix une confusion possible avec les lettres définies par le fabricant.

lettre/fonction	utilisation
C report (Carry)	additioneur binaire et décalage
L lien (Link)	unique. si jamais utilisé comme report
Z zéro (Zero)	à un: résultat est zéro
N signe (Negative)	à un: MSB à un (nombre négatif)
H report aux. (Half Carry)	report 4 bit (proc. 8 bits uniquement)
V dépassement (Overflow)	à un: dépassement de capacité pour des nombres arithmétiques
I interruption (Interrupt)	à un: interruption active
T traçage (Trace)	à un: mode de traçage actif
S superviseur (Supervisor)	à un: mode superviseur actif

Exemples: SET C, CLR C, etc.

OPERANDES

Opérande est le terme collectif pour différentes possibilités 'adressage: adresse effective, adressage immédiat, adressage de bit, liste, nombres arithmétiques et nombres point flottant.

La taille des données à transférer est donnée par un indicateur de données. L'indicateur de données est ajouté directement ou bien au code opératoire ou bien au(x) opérande(s).

Le diagrammes de syntaxe d'un opérande est donné à la fig. 24.

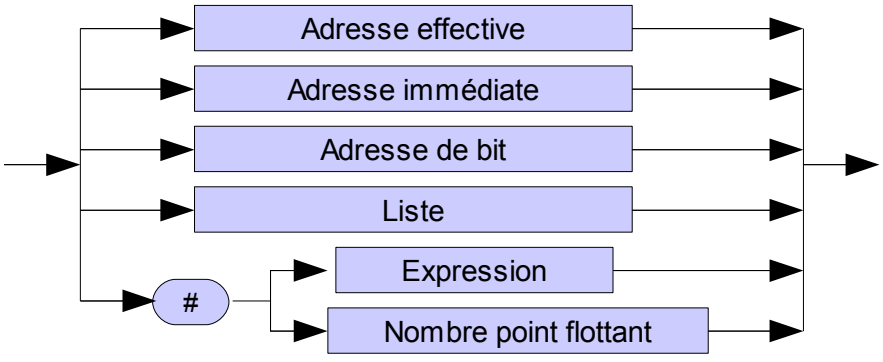


Fig. 24 opérande

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.

Fin du document.