Qu'est-ce qu'est CALM ?

CALM est l'abréviation pour Common Assembly Language for Microprocessors (langage d'assemblage commun pour microprocesseurs).

CALM n'est pas un nouveau langage de programmation mais plutôt une autre notation des instructions qui est plus uniforme et indépendante du fabricant. Actuellement, chaque fabricant définit pour son microprocesseur un propre langage d'assemblage. La terminologie utilisée dépend fortement du microprocesseur. Mais il manque toujours une notation uniforme pour des instructions qui sont fonctionnellement 100% identiques.

CALM profite du fait que beaucoup d'instructions exécutent exactement la même opération sur les microprocesseurs les plus différents. N'est-il donc pas dans ces cas évident de définir la même notation - indépendant du processeur - pour ces instructions?

Que définit CALM ?

CALM définit une syntaxe uniforme et indépendante du processeur pour les instructions et les pseudo-instructions. Suite à une longue expérience et des essais pratiques, on a pu exprimer presque toutes les instructions d'un microprocesseur avec les instructions définies par CALM et leurs notations.

CALM définit une terminologie d'assembleur cohérente. Et un concept est présenté qui montre, comment une instruction est composée. L'utilisateur comprend, pourquoi une instruction est écrite en CALM d'une certaine manière et pas d'une autre. En plus, une notation uniforme est définie pour les codes opératoires, les modes d'adressage, les indicateurs d'adresses et de données, etc.

CALM propose une notation unique des instructions pour tous les (micro)processeurs. CALM atteint cet objectif dans environ 95% des instructions d'un microprocesseur. Le reste est dû à des particularités spécifiques du processeur qui ne sont pas exprimables par un langage d'assemblage commun. Mais il est souvent dans ces cas préférable que ces particularités se distinguent aussi par une notation différente.

Quelles sont les avantages pour l'utilisateur ?

Une notation uniforme des instructions et particulièrement des modes d'adressages donne une image objective de la performance d'un microprocesseur. On peut donc comparer des microprocesseurs.

Pour la première fois, les programmeurs de différents microprocesseur peuvent communiquer ensemble. Jusqu'à présent, les notations différentes empêchaient ce premier pas.

Les changements de processeurs sont considérablement simplifiés, car la notation des instructions ne change pas. Il faut apprendre la structure générale et quelques particularités du nouveau processeur.

CALM n'est pas seulement utilisable pour les microprocesseurs mais aussi pour les miniprocesseurs, les processeurs des ordinateurs de grande taille et les unités microprogrammées. CALM est extensible et n'est pas limité qu'aux processeurs 8, 16 et 32 bits.

Et quelles sont les désavantages ?

CALM définit uniquement la partie logiciel d'un processeur. Si par contre quelqu'un a besoin de savoir des détails sur le matériel (temps d'exécution, code machine, brochage, charge électrique, indications concernant les fonctions intégrées comme compteur, unité d'accè à la mémoire [DMA], etc.), la documentation du fabricant est indispensable. L'utilisateur doit donc connaître temporairement les deux notations des instructions: celle de CALM et celle du fabricant.

Exemples

La page suivante illustre la notation des instructions en CALM et celle du fabricant pour les microprocesseurs 8080, iAPX86 et 68000.

```
; i8080: multiplication: RESULTAT.16 = MUL1.8 * MUL2.8
; modifie: A, B, D, E, H, L, F
MULT:
     MOVE   MUL1,A
     MOVE   A,E
     MOVE   #0,D
     MOVE   MUL2,A
     MOVE   #0,HL
     MOVE   #8,B
BOUCLE$:
     ADD    HL,HL
     RLC    A
     JUMP,CC SUITE$
     ADD    DE,HL
SUITE$:
     DEC    B
     JUMP,NE BOUCLE$
     MOVE   HL,RESULTAT
     RET
```

```
MULT:
     LDA  MUL1
     MOV  E,A
     MVI  D,0
     LDA  MUL2
     LXI  H,0
     MVI  B,8
BOUCLE:
     DAD  H
     RAL
     JNC  SUITE
     DAD  D
SUITE:
     DCR  B
     JNZ  BOUCLE
     SHLD RESULTAT
     RET
```

```
; iAPX86: traduit une chaîne de caractères EBCDIC en codes ASCII
; (<CR> termine); supposition: ES = DS, [DS] est équivalent à
; {DS}*16; modifie: AL, BX, CX, DI, SI, F
EBCDIC_ASCII:
     MOVE.16 #CONV_TAB,BX
     MOVE.16 #EBCDIC_CARACT,SI
     MOVE.16 #ASCII_CARACT,DI
     MOVE.16 [DS]+ASCII_LONGUEUR,CX
     AUTOINC
BOUCLE$:
     MOVE.8 [DS]+{SI!},AL
     MOVE.8 [DS]+{BX}+{AL},AL
     MOVE.8 AL,[ES]+{DI!}
     COMP.8 #16'D,AL
     LOOP,NE BOUCLE$
     RET.16
```

```
EBCDIC_ASCII:   PROC    NEAR
     MOV  BX,OFFSET CONV_TAB
     MOV  SI,OFFSET EBCDIC_CARACT
     MOV  DI,OFFSET ASCII_CARACT
     MOV  CX,SIZE ASCII_LONGUEUR
     CLD
BOUCLE:
     LODS EBCDIC_CARACT
     XLAT CONV_TAB
     STOS ASCII_CARACT
     CMP  AL,0DH
     LOOPNE BOUCLE
     RET          ; EQ: CR trouvé
```

```
; 68000: division: D4 = D5D4 / D3, reste dans D5, CS si erreur
DIV64:
     TEST.32 D3
     JUMP,ZS R8^DIV_ZERO$
     PUSH.32 D0
     MOVE.32 #32-1,D0
DIV_BOUCLE$:
     SETX
     RLX.32  D4
     RLX.32  D5
     JUMP,CS R8^DIV_DEPASSEMENT$
     SUB.32  D3,D5
     JUMP,HS R8^DIV_OK$
     ADD.32  D3,D5
     TCLR.32 D4:#0
DIV_OK$:
     DJ.16,NMO D0,DIV_BOUCLE$
     POP.32  D0
     CLRC
     RET
DIV_DEPASSEMENT$:
     POP.32  D0
DIV_ZERO$:
     SETC
     RET
```

```
DIV64      ; modifie: D3, D4, D5, F
     TST.L  D3
     BEQ.S  ZERO
     MOVE.L D0,-(A7)
     MOVEQ  #32-1,D0
BOUCLE
     ORI    #$10,CCR
     ROLX.L D4
     ROLX.L D5
     BCS.S  DEPASSEMENT
     SUB.L  D3,D5
     BCC.S  OK
     ADD.L  D3,D5
     BCLR   #0,D4
OK
     DBRA   D0,BOUCLE
     MOVE.L (A7)+,D0
     ANDI   #$FE,CCR
     RTS
DEPASSEMENT
     MOVE.L (A7)+,D0
ZERO
     ORI    #$1,CCR
     RTS
```

CALM Assembleur - description du produit (PC/MS-DOS, Atari ST, Smaky)

L'assembleur CALM est constitué de plusieurs programmes et fichiers:

Les programmes d'assembleur

ASCALM: Le véritable assembleur. Il fonctionne comme tous les autres assembleurs. Seule différence: Il faut écrire les programmes dans le langage d'assemblage CALM. Avantage: Ce langage d'assemblage ne se distingue pas de processeur à processeur. Avec le module correspondant, on peut générer un code machine (sans éditeur de liens) pour presque tous les microprocesseurs (voir liste des prix). Caractéristiques de l'assembleur: étiquettes (32 caractères significatifs), étiquettes locales, expressions avec une résolution de 32 bits, assemblage conditionnel (.IF/.ELSE/.ENDIF), listage conditionnel (.LIST/.ENDLIST), insertion de fichiers d'une taille quelconque (.INS), heure et date actuelle du système accessible par des constantes, insertion des messages d'erreurs directement dans le fichier source, générateur des références croisées, macros, etc. Caractéristiques (macro): on peut spécifier jusqu'à huit paramètres et un indicateur de données lors d'un appel de macro. La longueur d'un paramètre n'est limitée que par la longueur de ligne. On peut prédéfinir des paramètres. On peut distinguer au moment de l'appel à une macro entre les paramètres prédéfinis et transmis. Des macros peuvent appeler d'autres macros (jusqu'à 10 niveaux imbriqués). En plus, on peut analyser les paramètres transmis avec des fonctions spéciales (comparer, copier, tester, etc.). Ainsi on peut construire des macros puissantes qui p.ex. traduisent les instructions de la notation du fabricant en notation CALM. Format de sortie pour le code machine: MUFOM (voir sous MUFBIN).

MUFBIN: Convertit les fichiers objets (format MUFOM) générés par l'assembleur vers les formats: binaire (.BIN/.COM), hex, Intel hex (.HEX), Motorola format S (.FRS), PC/MS-DOS (.EXE) et Atari ST (.TOS/.TTP/.PRG).

Debogueur: Debogueur pour 8086 (DBGCALM, PC/MS-DOS) ou 68000 (DEBUG68, Atari ST/Smaky 100). Avec désassembleur utilisant la notation CALM.

Les programmes auxiliaires

CALMMENU	présente un menu simple.
FORMCALM	formate un fichier source (en notation CALM).
LSTTOASM	transforme un listage en source.
PFED	un éditeur de programme (avec macros!).
PROCSET	change la valeur par défaut dans les modules *.PRO.
SPACETAB	remplace les espaces par les tabulateurs.
TABSPACE	remplace les tabulateurs par les espaces.
TESTLIST	contrôle un fichier de listage.

Les fichiers pour un processeur

*.DOK	carte de réf. CALM pour le processeur *, p.ex. Z80.DOK.
*.PRO	le module pour le processeur *, par exemple Z80.PRO.
B*.TXT	description du module de processeur, p.ex. BZ80.TXT.
C*.TXT	comparaison des instr. (notation du fabricant -> CALM).
D*.EXE	disassembleur CALM pour le processeur *, p.ex. DZ80.EXE.
I*.TXT	liste des codes machines avec notation CALM (désassemblage).
ST*.ASM	liste triée des instructions en notation CALM.
S_*.ASM	exemples (ou E*.ASM ou *.ASM).
T*.ASM	fichier de test (liste des instructions).
xxx_CALM	traducteur (notation du fabricant -> CALM)
CALM_xxx	traducteur (CALM -> notation du fabricant)
Note:	selon le processeur, quelques fichiers ci-dessus manquent.

Remarques concernant l'assembleur CALM

Code objet sans éditeur de liens

L'assembleur CALM génère sans éditeur de liens (linker) un code objet dans le format dit MUFOM. Ainsi, l'utilisateur obtient après transformation du format MUFOM en format binaire un exécutable.

Dans la plupart des cas, ceci est suffisant. De plus, la combinaison assembleur-éditeur de liens qui génère des parties de programmes translatables, est souvent plus lente qu'un assembleur qui reassemble chaque fois le programme entier et qui génère directement le code machine. Mais pour cela, il faut qu'un matériel performant soit à disposition (p.ex. disques durs, RAM-disk).

La manière de programmation dépend des exigences. Pour les microprocesseurs (6502, 6800, 8080, Z80) et microordinateurs (sur une puce) simples 8 bits, les programmes sont relativement petits. Et ces processeurs sont souvent basés sur des systèmes d'exploitation qui chargent et exécutent les programmes toujours à la même adresse.

Les exigences sont plus sévères pour les microprocesseurs (6809, iAPX86, 68000, NS32000) et les systèmes d'exploitations plus performants. Les programmes doivent s'exécuter à n'importe quelle adresse mémoire et il faut diviser le programme en segments de programme, de données et de pile. Les deux exigences sont réalisables sans problème grâce aux modes d'adressage performants (adressage relatif, adressage indirect avec déplacement quelconque, etc.).

Avec l'assembleur CALM, le programmeur peut librement choisir le mode d'adressage. S'il veut générer des programmes indépendants de l'adresse mémoire (c.à.d. qui sont exécutables à n'importe quelle adresse sans relogement), il a seulement le droit d'utiliser l'adressage relatif. Il peut accéder les segments de données et de pile uniquement par l'adressage indirect. Récompense de ces limitations: le programme généré est directement exécutable à n'importe quelle adresse mémoire sans relogement.

Il faut aussi tenir en compte que l'environnement de programmation a changé. Personne n'accède aujourd'hui les programmes et les données par l'adressage absolu dans une mémoire centrale d'un Moctets et plus.

Ce concept demande donc une certaine discipline de la part du programmeur car il ne peut plus utiliser tous les modes d'adressage. Sinon, un assembleur avec éditeur de liens est nécessaire.

Code objet en format MUFOM

L'assembleur CALM génère un objet dans le format MUFOM. Ce format a quelques avantages par rapport aux formats .HEX (Intel) et format S (Motorola). En plus des caractéristiques des deux formats "standards" comme somme de contrôle, adresses de données, adresse de début, caractères ASCII, possibilité d'édition, etc., les informations suivantes s'y trouvent: version de l'assembleur et du module (description du processeur), nom du fichier, datation, type de processeur utilisé (.PROC), indications sur l'architecture du processeur, ainsi toutes les chaînes de caractères .TITLE et .CHAP apparaissent dans ce format. Toutes ces informations sont non-codées (caractères ASCII) et par conséquent lisibles par la commande TYPE.

En outre, le format MUFOM est aussi utilisable pour la génération d'objets translatables. Le format MUFOM est indépendant du processeur. Actuellement, l'assembleur CALM utilise uniquement les instructions MUFOM qui sont nécessaire pour des objets non-translatables.

Documentation du processeur

La documentation CALM livrée avec un processeur ne suffit pas pour comprendre un processeur dans tous les détails. Ceci est surtout vrai pour les microprocesseurs et microordinateurs (sur une puce) avec des fonctions intégrées (mémoire, accès direct à la mémoire [DMA], E/S). Il est donc presque toujours nécessaire de disposer de la documentation correspondante du fabricant. C'est pourquoi, la documentation CALM contient aussi des listes de comparaison.

Cartes de référence CALM

Carte de référence CALM

Dans une carte de référence CALM (en anglais CALM reference card), toutes les instructions d'un microprocesseur sont énumérées d'une manière bien disposée en utilisant le langage d'assemblage CALM qui est indépendant du fabricant. Sur quelques pages vous obtenez une vue d'ensemble étendue.

Intérêt d'une carte de référence CALM

Les cartes de référence CALM sont en premier lieu prévues comme aide à la programmation quotidienne: Quels modes d'adressage sont permis avec AND? Quels indicateurs sont modifiés avec COMP? etc.

Mais même si vous ne vous intéressez pas à l'assembleur CALM, une carte de référence peut être fort utile pour vous. Par exemple, si vous:

- voulez faire une meilleure connaissance de votre microprocesseur grâce à une présentation différente
- voulez obtenir une description indépendante du fabricant de toutes les instructions d'un microprocesseur
- voulez comparer des microprocesseurs et ne pas dépendre des indications du fabricant uniquement
- cherchez un nouveau, meilleur microprocesseur
- devez porter un jugement sur les performances d'un microprocesseur
- devez indiquer si un microprocesseur possède un(e) instruction/ code opératoire/mode d'adressage/type de données particulier
- voulez d'abord faire la connaissance de CALM

Organisation d'une carte de référence CALM

Toutes les cartes de référence CALM sont constituées de la même façon. Ceci donne d'une part un aspect homogène, et d'autre part, des comparaisons directes sont possibles.

De plus, les codes opératoires du fabricant sont indiqués en dessous des codes opératoires CALM.

Les cartes de référence CALM sont rédigées en anglais.

Livraison d'une documentation de carte de référence CALM

Une documentation de carte de référence CALM consiste en:

- carte de référence CALM
- comparaison des instructions: notation du fabricant -> notation CALM
- exemple (en notation CALM et du fabricant)
- une liste triée par ordre alphabétique de tous les codes opératoires (notation CALM)
- une liste triée par ordre alphabétique de tous les codes machines avec les instructions correspondantes (notation CALM)

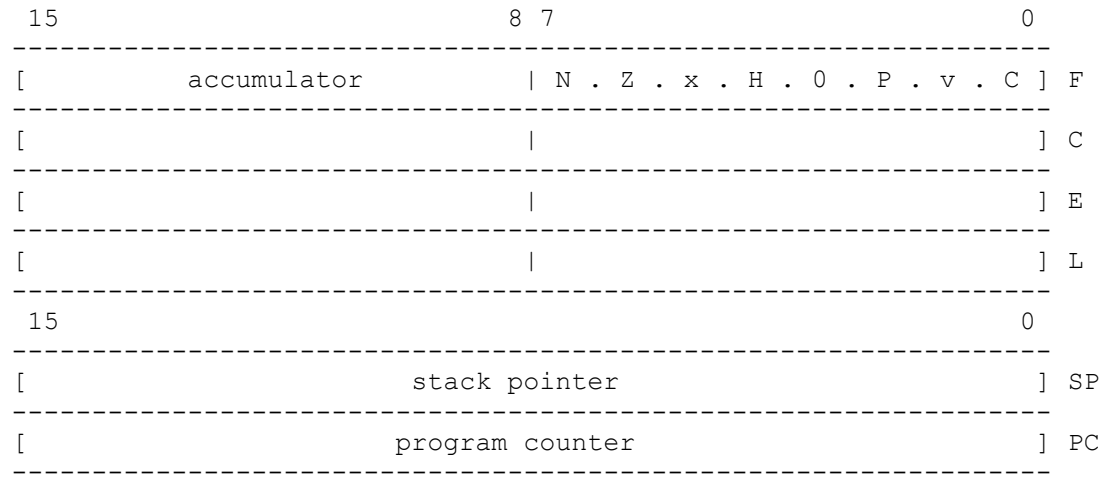
Exemple d'une carte de référence CALM

La carte de référence CALM du microprocesseur 8080/5 est présentée dans les pages suivantes.

8080/8085
CALM REFERENCE CARD

8080/8085 Description

Programming Model



General
Address: 16 bit
Data: 8 bit (8085: data multiplexed with addresses A0-A7)

Abbreviations used
v 16'0, 16'8, 16'10, 16'18, 16'20, 16'28, 16'30, 16'38
r8 A B C D E H L
s8 B C D E H L
r16 BC DE HL SP
i8 {BC} {DE} {HL}
VAL8 8-bit value
VAL16 16-bit value
cc EQ NE CS CC MI PL PO PE

Modifications versus CALM Standard
8 Bit: All transfers are 8 bits wide, except those determined by register names (1 letter = 8 bit, 2 letters = 16 bits).
Flag v Unspecified 8085 flag: 2's complement overflow (in arithmetic 8-bit and 16-bit operations). 8080: flag is always 1. (U8085)
Flag x Unspecified 8085 flag: sign(op1)*sign(op2) + sign(op1)*sign (result)(U8085) + sign(op2)*sign(result). For COMP and SUB, invert sign(op2). 8080: flag is always 0.

Remarks
- flag equalities: EQ=ZS, NE=ZC, CS=LO, CC=HS, MI=NS, PL=NC.
- Reset: IOFF
  JUMP 16'0
- Interrupt: IOFF
  CALL v
  Additional interrupt addresses for 8085: 16'2C, 16'34, 16'3C. (8085)
- NMI: IOFF (8085)
  CALL 16'24
- CALM - Intel register names: equal except: F=PSW and 16 bit names.
- CALM - Intel flag names: N=S, Z=Z, H=AC, P=P, C=C.

```
Transfer instructions
MOVE      #VAL8 |,A    []
          VAL16 |
          r8    |
          i8    |
          $VAL8 |
          A,| VAL16
            | r8
            | i8
            | $n
(MVI LDA MOV LDAX IN STA MOV STAX OUT)

MOVE      #VAL8 |,s8   []
          s8    |
          {HL}  |
          s8,{HL}
(MVI MOV MOV)

MOVE      #VAL16,r16   []
          VAL16,HL
          HL,VAL16
          HL,SP
(LXI LHLD SHLD SPHL)

MOVE      HL,{DE}      []       (U8085)
          {DE},HL
          #{HL}+VAL8,DE
          #{SP}+VAL8,DE
(SHLX LHLX LDHI LDSI)

PUSH |    r16          [], r16 without SP
POP  |    AF           [], [all] if POP AF
(PUSH POP)

SETC                  [C=1]
(STC)

EX        DE,HL       []
          {SP},HL
(XCHG XTHL)

Arithmetic instructions
ADD  |    #VAL8 |,A    [N,Z,H,P,C]
ADDC |    r8    |      [N,Z,H,P,C]
SUB  |    {HL}  |      [N,Z,H,P,C]
SUBC |                 [N,Z,H,P,C]
COMP |                 [N,Z,H,P,C]
(ADI ADD ACI ADC SUI SUB SBI SBB CPI CMP)

ADD       r16,HL      [C]
(DAD)

SUB       BC,HL       [N,Z,x,H,P,v,C] (U8085)
(DSUB)

INC |     r8          [N,Z,H,P]
DEC |     {HL}        [N,Z,H,P]
(INR DCR)

INC |     r16         []
DEC |
(INX DCX)
```

```
Logical instructions
AND |     #VAL8 |,A    [N,Z,H,P,C=0]
OR  |     r8    |      [N,Z,H,P,C=0]
XOR |     {HL}  |      [N,Z,H,P,C=0]
(ANA ANI ORA ORI XRA XRI)

NOT       A            []
NOTC                   [C]
(CMA CMC)

Shift instructions
RR  |     A            [C = A:#0]
RRC |                  [C = A:#0]
RL  |                  [C = A:#7]
RLC |                  [C = A:#7]
(RRC RAR RLC RAL)

ASR       HL           [C = L:#0] (U8085)
RLC       DE           [v,C = D:#7] (U8085)
(ARHL RDEL)

Program flow instructions
JUMP,cc | VAL16        []
JUMP    |
CALL,cc |
CALL    |
JUMP      {HL}         []
JUMP,XC | VAL16        []       (U8085)
JUMP,XS |
(J- JMP C- CALL PCHL JNX5 JX5)

RST       v            []
RST,VS    16'40        []       (U8085)
(RST RSTV)             one byte call (restart)

RET,cc                 []
RET                    []
WAIT                   []
NOP                    []
ION                    []
IOFF                   []
(R- RET HLT NOP EI DI)

Special instructions
DAA       A            [N,Z,H,P,C]
(DAA)                  Decimal Adjust A, only valid after ADD and ADDC

RIM |     A            []       (8085)
SIM |                  RIM: read interrupt mask
(RIM SIM)              SIM: set interrupt mask

RETEM                  [all]   return from emulation mode (V20)
(RETEM)                POP.16 IP; POP.16 CS; POP.16 SF;
                       MD bit write disable

TRAPNATIVE #VAL8       [MD=1]   trap to native mode (8086)(V20)
(CALLN)                PUSH.16 SF; PUSH.16 CS; PUSH.16 IP; SET MD;
                       return with RETI.32
Notes
(8085)    only available in 8085.
(U8085)   unspecified 8085 flag or operation code.
(V20)     only available in V20, V30, V40, and V50 (8080 emulation
          mode).                          (c) Patrick Faeh, June 1985.
```