INSTRUCTIONS

Une instruction est composée ou bien d'un code opératoire seulement ou bien d'un code opératoire suivi d'un ou plusieurs opérandes. Le code opératoire est un nom mnémotechnique dérivé d'un verbe anglais qui exprime l'opération à effectuer.

Des indicateurs de données supplémentaires fixent l'instruction dans tous ses détails.

Le diagramme de syntaxe général d'une instruction est illustré à la fig. 25.

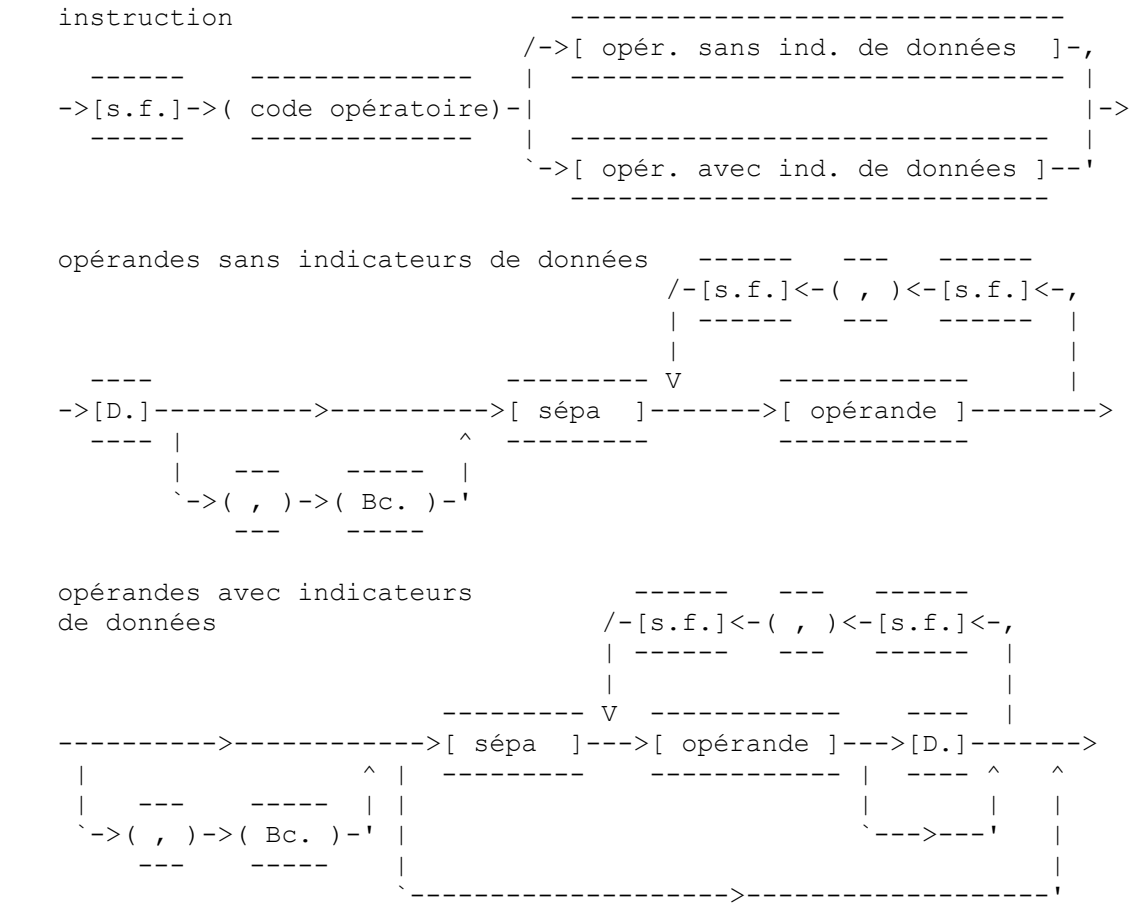
On peut ajouter des lettres à des codes opératoires CALM exprimant:

M instruction avec des opérandes multiples
P instruction provoquant une action pour des périphériques (E/S)
I instruction indivisible

La philosophie générale selon laquelle les indicateurs sont modifiés, varie fortement d'un processeur à un autre. Pour cette raison les indicateurs ne sont pas présentés dans les définitions des instructions suivantes.

Les instructions qui diffèrent de la philisophie générale du processeur, ne doivent pas avoir les noms définis par CALM.

On utilise dans le texte suivant l'abréviation .id pour indicateur de données.

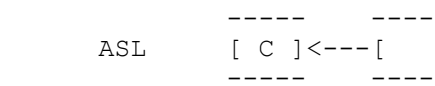


Bc.: code condition D.: indicateur de données

Fig. 25

```
ACOC          Add Complement, One and the Carry
   Descript.: Additionne le complément à un du premier opérande, un et
              l'indicateur de report C au second opérande. L'opération
              effectuée est équivalente à SUBC, mais les indicateurs
              sont modifiés autrement.
   Syntaxe:   ACOC.id source, sourcedestination
              ACOC.id source1, source2, destination
              ACOC    source.id, sourcedestination.id
              ACOC    source1.id, source2.id, destination.id
   Comment.:  Cette instruction existe seulement dans les anciens
              processeurs 8 bit (p.ex. SC/MP, 1802).
   Exemple:   ACO     #16'4A,D                ; 1802


ADD           ADD
   Descript.: Additionne deux opérandes et met à jour les indicateurs.
   Syntaxe:   ADD.id  source, sourcedestination
              ADD.id  source1, source2, destination
              ADD     source.id, sourcedestination.id
              ADD     source1.id, source2.id, destination.id
   Exemples:  ADD     {HL},A                  ; 8080, Z80
              ADD.16  {A6}-2,D0               ; 68000
              ADD.F64 {R0}+10,{R1}+20         ; NS32000


ADDC          ADD with Carry
   Descript.: Additionne deux opérandes et l'indicateur de report C.
   Syntaxe:   ADDC.id source, sourcedestination
              ADDC.id source1, source2, destination
              ADDC    source.id, sourcedestination.id
              ADDC    source1.id, source2.id, destination.id
   Exemples:  ADDC    {IX}+3,A                ; 6800, 6801
              ADDC.8  #1,[SS]+{BP}            ; iAPX86
              ADDX.D8 {-A0},{-A1}             ; 68000, X est ici C


AND           AND
   Descript.: Effectue un ET logique bit à bit des deux opérandes de
              source.
   Syntaxe:   AND.id  source, sourcedestination
              AND.id  source1, source2, destination
              AND     source.id, sourcedestination.id
              AND     source1.id, source2.id, destination.id
   Comment.:  Cette instruction sert à forcer à zéro (masquer) des
              bits. Une instruction similaire est BIC (bit clear),
              mais elle est rarement disponible.
   Exemples:  AND     #1,A                    ; 8080, Z80, ...
              AND.32  #16'FFFF0000,D6         ; 68000
              BIC.16  #7,R0            ; NS32000 (AND.16 #16'FFF8,R0)
```

ASL Arithmetic Shift Left



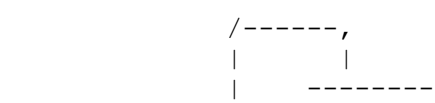
Descript.: Provoque un décalage de l'opérande de source de un ou plusieurs bits à gauche (direction MSB). Les bits des poids les plus faibles sont remplacés par des zéros. L'opération est équivalente à SL, mais l'indicateur de dépassement peut être modifié autrement. Si le signe du résultat est différent de celui de l'operérande de source, l'indicateur de dépassement doit être positionné.

Syntaxe:
```
ASL.id  sourcedestination
ASL.id  amplitude, sourcedestination
ASL.id  amplitude, source, destination
ASL     amplitude.id, sourcedestination.id
ASL     amplitude.id, source.id, destination.id
```

Comment.: Une amplitude négative signifie, que le décalage se fait à droite. Un décalage à gauche multiplie par deux. ASL.16 sourcedestination est équivalent à SL.A16 sourcedestination.

Exemple:
```
ASL.16  CL,BX                   ; iAPX86
```

ASR Arithmetic Shift Right



Descript.: Provoque un décalage de l'opérande de source de un ou plusieurs bits à droite (direction LSB), mais le bit de poids le plus fort (MSB, signe) est conservé et duplifié vers la droite. L'opération est équivalente à une division par deux pour des nombres positifs (les nombres négatifs ne sont pas arrondis dans la même direction).

Syntaxe:
```
ASR.id  sourcedestination
ASR.id  amplitude, sourcedestination
ASR.id  amplitude, source, destination
ASR     amplitude.id, sourcedestination.id
ASR     amplitude.id, source.id, destination.id
```

Comment.: Une amplitude négative signifie, que le décalage se fait à gauche. ASR.16 sourcedestination est équivalent à SR.A16 sourcedestination.

Exemples:
```
ASR     B                       ; 6809 (ASR #1,B)
ASR.16  CL,BX                   ; iAPX86
```

CALL CALL

Descript.: Sauve l'adresse de retour sur la pile et saute au sous-programme à l'adresse donnée.

Syntaxe:
```
CALL    adresse_de_saut
CALL,c  adresse_de_saut
```

Comment.: L'instruction RET termine le sous-programme.

Exemple:
```
CALL    ADRESSE
```

```
CHECK          CHECK
   Descript.:  Contrôle une valeur contre deux limites.
   Syntaxe:    CHECK.id    limite_inférieure, limite_supérieure, source
   Comment.:   Une double comparaison est effectuée. Une valeur limite
               est souvent la valeur zéro et n'est pas indiquée
               explicitement. Si la valeur ne se trouve pas à
               l'intérieur des limites, les processeurs réagissent
               assez différemment: ou bien uniquement un indicateur est
               positionné ou bien une interruption spéciale est
               provoquée (check trap).
   Exemple:    CHECK.A16 #100,D4              ; 68000 (0 <= D4 <= 100)


CLR            CLeaR
   Descript.:  Met à zéro l'opérande de destination (tous les bits sont
               mis à zéro). Instruction équival.:  MOVE #0,destination.
   Syntaxe:    CLR.id  destination
   Comment.:   Si une instruction est conditionelle, l'opérande n'est
               pas modifié si la condition est fausse (voir aussi SET).
   Exemple:    CLR     {IX}+DISTANCE:#BIT_NR  ; Z80
               CLR.32  D1                     ; 68000
               CLRC


COMP           COMPare
   Descript.:  Compare le second opérande au premier. C'est la même
               opération que SUB, mais il n'y a pas de résultat généré
               (uniquement les indicateurs sont mis à jour).
   Syntaxe:    COMP.id source1, source2
               COMP    source1.id, source2.id
   Exemples:   COMP      {IX}+10,B            ; 6800, ...
               COMP.F32  {SB}+4,F0            ; NS32000


CONV           CONVert
   Descript.:  Effectue une conversion de type de données. Si la source
               et la destination sont identiques, les indicateurs de
               données peuvent être ajoutés après le code opératoire.
   Syntaxe:    CONV.id source, destination
               CONV    source.id, destination.id
   Comment.:   Il faut spécifier un type de données et une taille de
               données. Il est possible de remplacer quelques
               conversions de type de données par l'instruction MOVE.
   Exemples:   CONV.A8.16  D0                 ; 68000
               CONV        {FP}.A16,{SB}.F64  ; NS32000


DEC            DECrement
   Descript.:  Décrémente l'opérande de un. Uniquement un opérande est
               admis. Il faut utiliser une instruction SUB pour des
               décréments de 2, 4, etc.
   Syntaxe:    DEC.id  sourcedestination
   Exemples:   DEC     {IX}+DEPLACEMENT       ; Z80
               DEC.32  D1                     ; 68000
```

DIV DIVide

Descript.: Divise le second opérande par le premier. La destination est soit le second ou un troisième opérande. Le reste peut être un dernier opérande additionnel.

Syntaxe: DIV.id diviseur, dividende, quotient, reste
DIV diviseur.id,dividende.id,quotient.id,reste.id

Comment.: Selon la taille des opérandes un dépassement de capacité peut se produire. Ceci génère automatiquement une interruption (trap) dans quelques processeurs. L'opérande de destination est souvent identique avec un des opérandes de source et contient parfois aussi le reste de la division dans sa partie supérieure.

Exemples:
```
DIV.16  CX,DXAX,AX,DX           ; iAPX86
DIV.A16 D1,D0                ; 68000 (reste: D0:#31..#16)
```
Il faut que: 5.DIV.2 = 2, -5.DIV.2 = -2, 5.DIV.-2 = -2, -5.DIV.-2 = 2. Et: A = (A.DIV.B)*B + (A.MOD.B), .MOD. = reste de la division.

DJ Decrement and Jump

Descript.: Décrémente le premier opérande et saute à l'adresse donnée par le second opérande.

Syntaxe: DJ.id,c sourcedestination, addresse_de_saut

Comment.: L'instruction de boucle la plus utilisée décrémente un compteur de un et saute à l'adresse donnée si une condition est satisfaite.

Exemples:
```
DJ,NE     B,ADRESSE             ; Z80
DJ.16,NMO D0,ADRESSE         ; 68000 (NMO: pas moins un)
```

EX EXchange

Descript.: Echange deux opérandes. L'ordre des deux opérandes ne joue aucun rôle.

Syntaxe: EX.id opérande1, opérande2

Comment.: L'instruction EX effectue simultanément une double instruction MOVE.

Exemples:
```
EX      {SP},HL                 ; 8080, Z80
EX.16   AX,[DS]+SEMA            ; iAPX86
EX.32   A6,D0                   ; 68000
```

HALT HALT

Descript.: Arrête le processeur jusqu'à ce qu'il soit réinitialisé par un signal externe.

Syntaxe: HALT

Comment.: Le processeur libère après cette instruction le bus et ne réagit même plus aux requêtes d'interruption. On peut réactiver le processeur seulement par un signal externe de réinitialisation. Cette instruction est rare. Par contre, l'état HALT est courant sur tous les microprocesseurs modernes (p.ex. double erreur de bus).

Exemple:
```
HALT                            ; PDP-11
```

INC INCrement

Descript.: Incrémente l'opérande de un. Uniquement un opérande est admis. Il faut utiliser l'instruction ADD pour des incréments de 2, 4, etc.

Syntaxe: INC.id sourcedestination

Exemples:
```
INC     {IX}+DISTANCE           ; Z80
INC.32  D1                      ; 68000
```

```
IOFF          Interrupt OFF
   Descript.: Interdit des interruptions.
   Syntaxe:   IOFF
   Comment.:  L'instruction IOFF interdit tous les interruptions sauf
              l'interruption non masquable (NMI) qui est toujours
              possible.
   Exemples:  IOFF                          ; 8080, Z80, 6800, ...
              IOFF                        ; 68000 (OR.16 #16'0700,SF)


ION           Interrupt ON
   Descript.: Permet des interruptions.
   Syntaxe:   ION
   Exemples:  ION                           ; 8080, Z80, 6800, ...
              ION                       ; 68000 (AND.16 #16'F8FF,SF)


JUMP          JUMP
   Descript.: Saute à l'adresse donnée.
   Syntaxe:   JUMP    adresse_de_saut
              JUMP,c  adresse_de_saut
   Comment.:  L'instruction  JUMP ADRESSE  est équivalente à
              MOVE #ADRESSE,PC.
   Exemples:  JUMP,VS ADRESSE               ; Z80, 6800, ...
              JUMP    R16^ADRESSE           ; 6809, iAPX86, ...
              JUMP    {{FP}+DISTANCE1}+DISTANCE2 ; NS32000


MOVE          MOVE
   Descript.: Transfère l'opérande de source dans l'opérande de
              destination.
   Syntaxe:   MOVE.id source, destination
              MOVE    source.id, destination.id
   Exemples:  MOVE    HL,SP                ; 8080, Z80
              MOVE.8  #2,BL                ; iAPX86
              MOVE.32 #{A6}+DISTANCE,A0    ; 68000
              MOVE.F64 {SB}+DISTANCE,F2     ; NS32000


MUL           MULtiply
   Descript.: Multiplie les deux opérandes de source. Si l'opérande de
              destination n'est pas explicitement précisé par sa
              taille de données, il n'est pas clair, si la taille de
              l'opérande de destination est simple ou double.
   Syntaxe:   MUL.id  source, sourcedestination
              MUL.id  source1, source2, destination
              MUL     source.id, sourcedestination.id
              MUL     source1.id, source2.id, destination.id
   Exemples:  MUL     A,B,AB                ; 6801
              MUL.16  D1,D0           ; 68000 (MUL D1.16,D0.16,D0.32)
              MUL.F64 {R0}+10,F2            ; NS32000


NEG           NEGate
   Descript.: Calcule le complément vrai (complément à deux ou
              soustraction de zéro) de l'opérande de source.
   Syntaxe:   NEG.id  sourcedestination
              NEG.id  source, destination
              NEG     source.id, destination.id
   Exemples:  NEG     A                     ; Z80, ...
              NEG.16  {A6+}                 ; 68000
              NEG.F64 {R0}+10,{R1}+20       ; NS32000
```

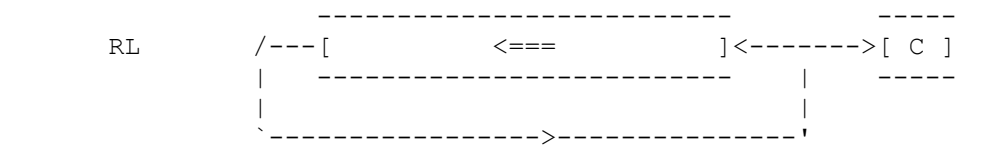
```
NEGC          NEGate with Carry
   Descript.: Calcule le complément vrai avec l'indicateur de report
              C, c.à.d. soustrait l'opérande et l'indicateur de report
              C de zéro.
   Syntaxe:   NEGC.id sourcedestination
              NEGC.id source, destination
              NEGC    source.id, destination.id
   Comment.:  Le résultat d'une opération NEGC est identique avec
              celui d'une opération NOT si l'indicateur de report C a
              été positionné avant l'opération. L'état final de
              l'indicateur de report C peut par contre être différent.
   Exemple:   NEGX.D8 D0                      ; 68000 (X est ici C)


NOP           No OPeration
   Descript.: Cette instruction ne fait rien.
   Syntaxe:   NOP
   Comment.:  Le microprocesseur cherche l'instruction de la mémoire
              mais n'effectue pas d'opération.
   Exemple:   NOP


NOT           NOT
   Descript.: Inverse chaque bit de l'opérande source (complément à
              un).
   Syntaxe:   NOT.id  sourcedestination
              NOT.id  source, destination
              NOT     source.id, destination.id
   Exemples:  NOT     A                       ; 8080, Z80, ...
              NOT.16  {A6+}                   ; 68000
              NOT.32  {R0}+10,{R1}+20         ; NS32000


OR            OR
   Descript.: Effectue un OU inclusif bit à bit des deux opérandes
              de source.
   Syntaxe:   OR.id   source, sourcedestination
              OR.id   source1, source2, destination
              OR      source.id, sourcedestination.id
              OR      source1.id, source2.id, destination.id
   Comment.:  Cette instruction sert à forcer à un des bits.
              L'instruction OR est quelquefois appelée BIS (bit set),
              particulièrement si l'instruction BIC (bit clear) est
              disponible.
    Exemples: OR      #1,A                    ; 8080, Z80, ...
              OR.32   #16'FFFF0000,D6         ; 68000
              BIS.8   #3,F                    ; NS32000 (OR.8 #16'3,F)


POP           POP
   Descript.: Dépiler. Pour la plupart des processeurs on a: POP
              destination  est équivalent à   MOVE {SP+},destination.
   Syntaxe:   POP.id  destination
   Comment.:  Normalement, uniquement une pile reçoit l'adresse de
              retour d'un sous-programme (CALL). L'instruction POP se
              réfère à cette pile.
   Exemples:  POP     DE                      ; 8080, Z80
              POPU    AB                      ; 6809 (de la pile US)
              POP.16  F                       ; 68010, 68020
```

PUSH PUSH
Descript.: Empiler. Pour la plupart des processeurs on a:
PUSH source est équivalent à MOVE source,{-SP}
Syntaxe: PUSH.id source
Comment.: Normalement, uniquement une pile reçoit l'adresse de retour d'un sous-programme (CALL). L'instruction PUSH se réfère à cette pile.
Exemples:
```
PUSH    DE                      ; 8080, Z80
PUSHU   AB                      ; 6809 (de la pile US)
PUSH.16 F                       ; 68010, 68020
```

RESET RESET
Descript.: Réinitialise les périphériques.
Syntaxe: RESET
Comment.: L'instruction RESET simule le signal de réinitialisation (RESET) par programmation (la ligne RESET est bidirectionelle). Le processeur lui-même n'est pas réinitialisé.
Exemple:
```
RESET                           ; 68000
```

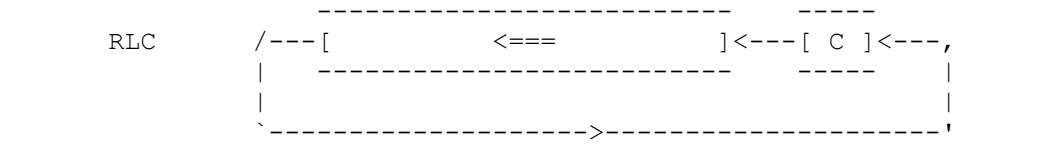
RET RETurn
Descript.: Saute à l'adresse récupérée de la pile.
Syntaxe:
```
RET
RET,c
RET     expression
```
Comment.: L'instruction RET termine un sous-programme appelé par CALL. A la fin des sous-programmes d'interruption ou d'exception (trap), différentes instructions RET signalent qu'il y a d'autres informations à récupérer de la pile (en plus du compteur d'adresses PC).
Exemples:
```
RET,EQ                          ; 8080, Z80
RETI                            ; Z80 (interruption)
RETSF                         ; 68000 (PC + registre SF)
RET     #10                     ; NS32000
```

RL Rotate Left



Descript.: Effectue un décalage de l'opérande de source d'un ou plusieurs bits à gauche. A chaque décalage, le LSB est remplacé par le MSB.
Syntaxe:
```
RL.id   sourcedestination
RL.id   amplitude, sourcedestination
RL.id   amplitude, source, destination
RL      amplitude.id, sourcedestination.id
RL      amplitude.id, source.id, destination.id
```
Comment.: Une amplitude négative signifie, que le décalage se fait à droite.
Exemples:
```
RL      B                       ; 6809 (RL #1,B), ...
RL.16   CL,BX                   ; iAPX86 (RL CL.8,BX.16)
```

RLC Rotate Left with Carry



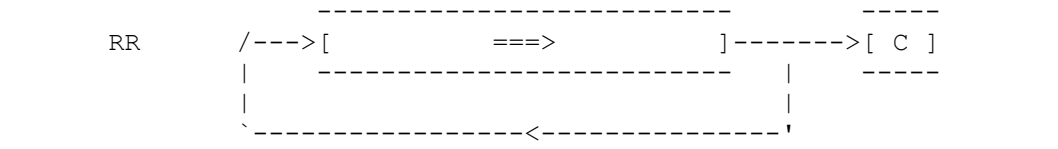
Descript.: Effectue un décalage de l'opérande de source d'un ou plusieurs bits à gauche. A chaque décalage, le LSB est remplacé par l'indicateur de report C et l'indicateur de report C par le MSB.

Syntaxe:
```
RLC.id  sourcedestination
RLC.id  amplitude, sourcedestination
RLC.id  amplitude, source, destination
RLC     amplitude.id, sourcedestination.id
RLC     amplitude.id, source.id, destination.id
```

Comment.: Une amplitude négative signifie, que le décalage se fait à droite.

Exemples:
```
RLC     B                       ; 6809 (RLC #1,B), ...
RLC.16  CL,BX                   ; iAPX86 (RLC CL.8,BX.16)
```

RR Rotate Right



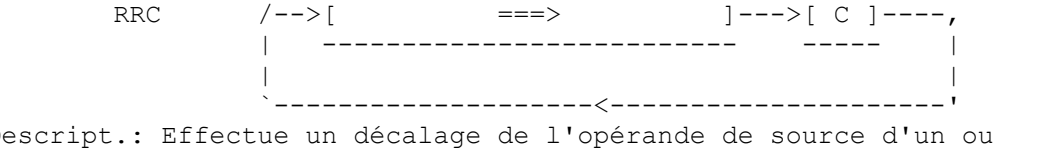
Descript.: Effectue un décalage de l'opérande de source d'un ou plusieurs bits à droite. A chaque décalage, le MSB est remplacé par le LSB.

Syntaxe:
```
RR.id   sourcedestination
RR.id   amplitude, sourcedestination
RR.id   amplitude, source, destination
RR      amplitude.id, sourcedestination.id
RR      amplitude.id, source.id, destination.id
```

Comment.: Une amplitude négative signifie, que le décalage se fait à gauche.

Exemples:
```
RR      B                       ; 6809 (RR #1,B), ...
RR.16   CL,BX                   ; iAPX86 (RR CL.8,BX.16)
```

RRC Rotate Right with Carry



Descript.: Effectue un décalage de l'opérande de source d'un ou plusieurs bits à droite. A chaque décalage, le MSB est remplacé par l'indicateur de report C et l'indicateur de report C par le LSB.

Syntaxe:
```
RRC.id  sourcedestination
RRC.id  amplitude, sourcedestination
RRC.id  amplitude, source, destination
RRC     amplitude.id, sourcedestination.id
RRC     amplitude.id, source.id, destination.id
```

Comment.: Une amplitude négative signifie, que le décalage se fait à gauche.

Exemples:
```
RRC     B                       ; 6809 (RRC #1,B), ...
RRC.16  CL,BX                   ; iAPX86 (RRC CL.8,BX.16)
```

SET SET

Descript.: Initialise l'opérande de destination avec des uns (tous les bits sont mis à un). Instruction équivalente: MOVE #-1,destination.

Syntaxe: SET.id destination

Comment.: Si une instruction SET est conditionelle, l'opérande est initialisé avec des uns si la condition est vraie, et initialisé avec des zéros si la condition est fausse (voir aussi CLR).

```
Exemples: SET.8,EQ  {A0}                ; 68000
          SET       {HL}:#4             ; Z80
          SETC                          ; 8080, Z80, ...
```

SKIP SKIP

Descript.: Saute par dessus l'instruction suivante.

Syntaxe: SKIP,c

Comment.: L'instruction SKIP est pratiquement toujours combinée avec une condition.

```
Exemple:  SKIP,EQ DJ.16,NMO D0,ADRESSE    ; 68000
```

SL Shift Left

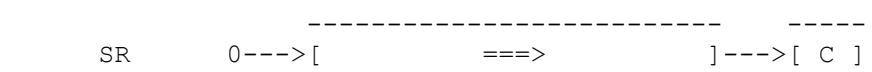
Descript.: Effectue un décalage de l'opérande de source d'un ou plusieurs bits à gauche. A chaque décalage, le LSB est remplacé par un zéro.

Syntaxe:
```
SL.id   sourcedestination
SL.id   amplitude, sourcedestination
SL.id   amplitude, source, destination
SL      amplitude.id, sourcedestination.id
SL      amplitude.id, source.id, destination.id
```

Comment.: Une amplitude négative signifie, que le décalage se fait à droite. Un décalage à gauche multiplie par deux.

```
Exemples: SL      B                     ; 6809 (SL #1,B), ...
          SL.16   CL,BX                 ; iAPX86 (SL CL.8,BX.16)
```

SR Shift Right



Descript.: Effectue un décalage de l'opérande de source d'un ou plusieurs bits à droite. A chaque décalage, le MSB est remplacé par un zéro.

Syntaxe:
```
SR.id   sourcedestination
SR.id   amplitude, sourcedestination
SR.id   amplitude, source, destination
SR      amplitude.id, sourcedestination.id
SR      amplitude.id, source.id, destination.id
```

Comment.: Une amplitude négative signifie, que le décalage se fait à gauche. Un décalage à droite divise par deux.

```
Exemples: SR      B                     ; 6809 (SR #1,B), ...
          SR.16   CL,BX                 ; iAPX86 (SR CL.8,BX.16)
```

```
SUB          SUBtract
   Descript.: Soustrait le premier opérande du second opérande.
   Syntaxe:   SUB.id  source, sourcedestination
              SUB.id  source1, source2, destination
              SUB     source.id, sourcedestination.id
              SUB     source1.id, source2.id, destination.id
   Exemples:  SUB     #2'1010,A              ; 8080, Z80, ...
              SUB.32  {A6}+400,D6            ; 68000
              SUB.F64 F0,{SB}+10             ; NS32000


SUBC         SUBtract with Carry
   Descript.: Soustrait le premier opérande et l'indicateur de report
              C du second opérande.
   Syntaxe:   SUBC.id source, sourcedestination
              SUBC.id source1, source2, destination
              SUBC    source.id, sourcedestination.id
              SUBC    source1.id, source2.id, destination.id
   Exemples:  SUBC     #2'1010,A              ; 8080, Z80, ...
              SUBX.32  {-A4},{-A3}            ; 68000 (X est ici C)
              SUBC.32  R0,{SB}+10             ; NS32000


SWAP         SWAP
   Descript.: Echange les deux moitiés de l'opérande. La taille
              associée se réfère à la taille entière de l'opérande.
   Syntaxe:   SWAP.id sourcedestination
   Comment.:  L'instruction SWAP est nécessaire, si l'on peut pas
              accéder indépendamment aux sous-unités d'un opérande.
   Exemple:   SWAP.32 D4                     ; 68000


TCLR         Test and CLeaR
   Descript.: Teste et met à zéro le(s) bit(s) d'opérande.
   Syntaxe:   TCLR.id sourcedestination    (:adresse_de_bit(.id))
              TCLR    sourcedestination.id (:adresse_de_bit(.id))
   Comment.:  L'indicateur de nullité Z contient souvent la valeur du
              bit avant l'exécution de l'instruction. La lettre
              supplémentaire I signale que l'instruction est
              indivisible. L'indicateur de données, ajouté au code
              opératoire ou à l'opérande d'adresse d'octet, indique
              que l'adressage de bit est limité.
   Exemples:  TCLR.32   D5:#31               ; 68000 (bit: 0..31)
              TCLR.8    {A6}+CONDITION:#2    ; 68000 (bit: 0..7)
              TCLRI     {SB}:{FP}.A8       ; NS32000 (bit: -128..+127)


TEST         TEST
   Descript.: Teste le signe et la valeur (si zéro) de l'opérande.
   Syntaxe:   TEST.id      source      (:adresse_de_bit(.id))
              TEST         source.da   (:adresse_de_bit(.id))
              TEST.id      source1, source2
   Comment.:  TEST opérande  est équivalent à  COMP #0,opérande.
              L'indicateur de nullité Z est positionné si la valeur
              de l'opérande est zéro. Le MSB est copié dans
              l'indicateur de signe N. Avec deux opérandes, une
              opération AND est effectuée avant le test.
   Exemples:  TEST      B                    ; 6800, ...
              TEST.32   D5:#31               ; 68000 (bit: 0..31)
              TEST.8    {A6}+CONDITION:#2    ; 68000 (bit: 0..7)
              TEST      {SB}:{FP}.A8       ; NS32000 (bit: -128..+127)
              TEST.16   [DS]+{SI},AX         ; iAPX86
```

```
TNOT          Test and NOT
   Descript.: Teste et invertit le(s) bit(s) d'opérande.
   Syntaxe:   TNOT.id sourcedestination    (:adresse_de_bit(.id))
              TNOT    sourcedestination.id (:adresse_de_bit(.id))
   Comment.:  L'indicateur de nullité Z contient souvent la valeur du
              bit avant l'exécution de l'instruction. L'indicateur de
              données après le code opératoire ou l'opérande d'adresse
              d'octet indique que l'adressage de bit est limité.
   Exemples:  TNOT.32   D5:#31                ; 68000 (bit: 0..31)
              TNOT.8    {A6}+CONDITION:#2     ; 68000 (bit: 0..7)
              TNOT      {SB}:{FP}.A8       ; NS32000 (bit: -128..+127)


TRAP          TRAP
   Descript.: Appel de sous-programme spécial.
   Syntaxe:   TRAP    expression
              TRAP,c  expression
   Comment.:  Les instructions TRAP peuvent être déclenchées par
              programmation ou par matériel (p.ex. division par zéro).
   Exemples:  TRAP                            ; 6800, ...
              TRAP    #15                     ; 68000
              TRAP,VS                         ; 68000


TSET          Test and SET
   Descript.: Teste et met à un le(s) bit(s) d'opérande.
   Syntaxe:   TSET.id sourcedestination    (:adresse_de_bit(.id))
              TSET    sourcedestination.id (:adresse_de_bit(.id))
   Comment.:  L'indicateur de nullité Z contient souvent la valeur du
              bit avant l'exécution de l'instruction. La lettre
              supplémentaire I signale que l'instruction est
              indivisible. L'indicateur de données, ajouté après le
              code opératoire ou l'opérande d'adresse d'octet, indique
              que l'adressage de bit est limité.
   Exemples:  TSET.32   D5:#31                ; 68000 (bit: 0..31)
              TSET.8    {A6}+CONDITION:#2     ; 68000 (bit: 0..7)
              TSETI     {SB}:{FP}.A8       ; NS32000 (bit: -128..+127)


WAIT          WAIT
   Descript.: Attend sur une interruption.
   Syntaxe:   WAIT
   Comment.:  L'instruction WAIT minimise le temps de réponse à une
              requête d'interruption. Après une interruption et après
              l'exécution du sous-programme d'interruption, le
              processeur exécute l'instruction suivante (après WAIT).
              Il y a quatre variantes du WAIT: WAIT (CPU attend
              activement), SLEEP (le courant est coupé des E/S), STOP
              (l'oscillateur interne est arrêté) et HALT (CPU
              s'arrête).
   Exemple:   WAIT

XOR           XOR
   Descript.: Effectue un OU exclusif bit à bit des deux opérandes de
              source.
   Syntaxe:   XOR.id  source, sourcedestination
              XOR.id  source1, source2, destination
              XOR     source.id, sourcedestination.id
              XOR     source1.id, source2.id, destination.id
   Comment.:  Cette instruction sert à invertir des bits.
   Exemples:  XOR     #1,A                    ; 8080, Z80, ...
              XOR.32  #16'FFFF0000,D6         ; 68000
```