Résumé Assembleur

Labo 5

18.11.2020

Owen Gombas
ISC1C

bra Entry

Additions et soustractions 16 bits

ADDD adresse (ADDD #23)

addition du contenu d'une case mémoire (16 bits) et D avec résultat dans D

SUBD adresse (SUBD $1002)

soustraction du contenu d'une case mémoire (16 bits) à D avec résultat dans D

- Pour ces deux instructions les modes d'adressage autorisés sont:
  - immédiat
  - direct
  - étendu
  - tous les indexés
- Affectent le CCR
- Il est plus simple d'utiliser ces instructions pour les additions/soustractions 16 bits que les instructions 8 bits avec retenue.
- Il n'y a pas d'instructions additions/soustractions 16 bits avec retenue.
  Il faut alors utiliser les instructions 8 bits pour faire des additions/soustractions supérieures à 16 bits.

Incréments et décréments

INC adresse

Additionne 1 à une case mémoire de 8 bits

DEC adresse

Enlève 1 à une case mémoire de 8 bits

INCA

Additionne 1 à A

DECA

Additionne 1 à A

INCB

Additionne 1 à B

DECB

Additionne 1 à B

INS

Additionne 1 à SP

DES

Additionne 1 à SP

INX

Additionne 1 à X

DEX

Additionne 1 à X

INY

Additionne 1 à Y

DEY

Additionne 1 à Y

- Ces instructions affectent le CCR (sauf le Carry)
- Il n'y pas d'instructions pour D
  on peut utiliser ADDD et SUBD

Multiplications 8bits

MUL

Multiplication de 2 nombres 8 bits non signés, situés dans A et B avec le résultat sur 16 bits dans D

- Fonctionne de façon incorrecte avec des nombres signés
- Aussi rapide qu'une addition
- Modifie le carry si le résultat indique le bit 7 de B à 1.

  Utilisé Pour les calculs fractionnaires
  les 8 bits de A étant la partie entière et les 8 bits de B la partie fractionnaire
  le carry indique donc que la partie fractionnaire ≥ 0,5.
  Dans certains cas le carry nous permet d'arrondir la partie entière. (mul puis adca #0)

  Par exemple sur 8 bits :

2^3	2^2	2^1	2^0		$0,5 = 2^{-1}$	$0,25 = 2^{-2}$	$0,125 = 2^{-3}$	$0,0625 = 2^{-4}$
1	0	1	1	.	1	0	1	1

Multiplication 16 bits

EMUL

Multiplication non signée de D par Y avec résultat dans Y:D

EMULS

Multiplication signée de D par Y avec résultat dans Y:D

- Les deux instructions génèrent un produit de 32 bits avec le MSB dans Y et le LSB dans D

Y	D
$1A32	$2B4A

- Modifient le CCR selon le résultat de l'opération (N, Z et C)
  Le carry est à 1 si le bit 15 de D est à 1.
  (Utilisé pour les calculs fractionnaires -> arrondi).

```
        org $1000    ; Zone de donnée en RAM
A1: ds 2             ; Déclaration d'une variable de 2 bytes A1
C1: ds 2             ; Déclaration d'une variable de 2 bytes C1
E1: ds 4             ; Déclaration d'une variable de 4 bytes E1
        org $2000    ; Zone de code
ldy A1               ; Charge A1 dans Y
ldd C1               ; Charge C1 dans D
emul                 ; Multiplie A1 par C1 avec le résultat dans Y:D
sty E1               ; Sauvegarde des 16 bits de poids fort
std E1+2             ; Sauvegarde des 16 bits de poids faible
```

Divisions 16 bits

IDIV

Division non signée de D par X avec le quotient dans X et le reste dans D

- Pour une division par 0, donne un quotient de $FFFF et le carry à 1.
- Met le flag Z à 1 si le quotient est nul

IDIVS

Division signée de D par X avec le quotient dans X et le reste dans D

- Met le flag V à 1 avec une division (-32768) / (-1) = 32768 (>+32767).
- Modifie aussi les drapeaux N et Z selon le résultat de la division

```
; Calcul de la valeur moyenne de 3 nombres sans tenir compte des
; dépassements
ldd    V1              ; V1 dans D
addd   V2              ; Addition de V2 à V1
addd   V3              ; Addition de V3 à V2+V1
ldx    #3              ; Le diviseur dans X
idiv                   ; Division
stx    AVERAGE         ; Sauvegarde du quotient (donc la moyenne) dans AVERAGE
```

Divisions 32 bits

EDIV

Division non signée de Y:D par X avec le quotient dans Y et le reste dans D

- V est à 1 si le quotient est plus grand que $FFFF et C est à 1 si le diviseur est à 0.

EDIVS

Division signée de Y:D par X avec le quotient dans Y et le reste dans D

- V est à 1 si le quotient est plus grand que $7FFF ou plus petit que $8000
- C est à 1 si le diviseur est 0

```
; Calcul de la moy. de 3 nombres de 16 bits en tenant compte des retenues
ldd    V1      ; V1 dans D
ldy    #0      ; Effacement de Y (MSW du nombre à diviser)
addd   V2      ; Addition de V2 à V1
exg    Y,D     ; Echange de Y et D
adcb   #0      ; Additionne le carry à B (qui contient Y donc le MSW)
exg    Y,D     ; Echange de Y et D
addd   V3      ; Additionne V3 à V1+V2
exg    Y,D     ; Echange de Y et D
adcb   #0      ; Additionne le carry à B (qui contient Y donc le MSW)
exg    Y,D     ; Echange de Y et D
ldx    #3      ; Charge X avec le diviseur
ediv           ; Division
sty    AVERAGE      ; Sauvegarde du quotient dans AVERAGE
```