Résumé Assembleur

Labo 6

19.11.2020

Owen Gombas
ISC1C

bra Entry

Décalage arithmétique
(tiens compte du bit de signe)

Décalage arithmétique à droite

ASR adresse

décalage arithmétique d'un bit vers la droite d'un byte en mémoire

ASRA

décalage arithmétique d'un bit vers la droite de A

ASRB

décalage arithmétique d'un bit vers la droite de B

- Décale chaque bit de 1 sur la droite
- Remplis avec des 0 si le bit de poids fort est à 0
- Remplis avec des 1 si le bit de poids fort est à 1
- Etat du bit 0 est copié dans le carry
- Est en réalité une division de $2^{nb\ de\ décalage}$
- Le carry nous indique s'il y'a eu un reste à la division
- Le mode d'adressage étendu et tout les indexés fonctionnent

Décalage arithmétique à gauche

ASL adresse

décalage arithmétique d'un bit vers la gauche d'un byte en mémoire

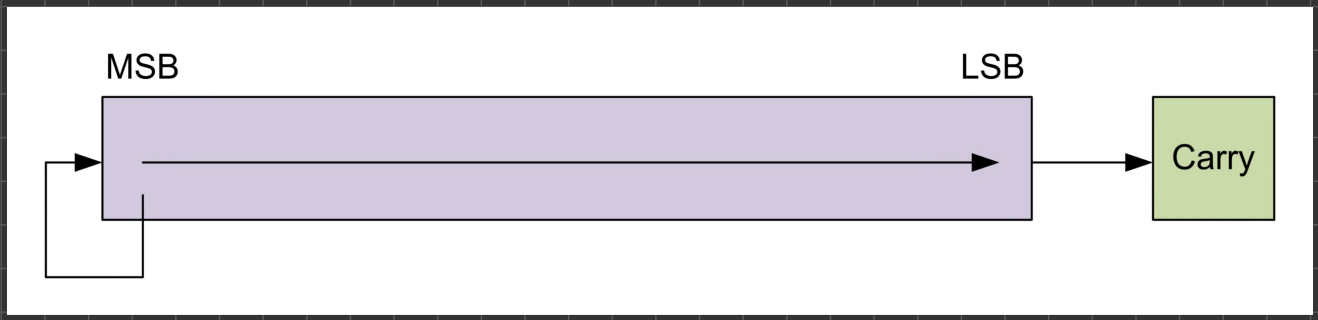
ASLB

décalage arithmétique d'un bit vers la gauche de A

ASLB

décalage arithmétique d'un bit vers la gauche de B

- Décale chaque bit de 1 sur la gauche
- Remplis avec des 0 si le bit de poids fort est à 0
- Remplis avec des 1 si le bit de poids fort est à 1
- Etat du bit 0 est copié dans le carry
- Est en réalité une division de $2^{nb\ de\ decalage}$
- Le carry nous indique s'il y'a eu un reste à la division
- Le mode d'adressage étendu et tout les indexés fonctionnent



Décalages logiques
(ne tient pas compte du bit de signe)

Décalage logique à droite

LSR adresse

décalage logique à droite d'un byte en mémoire

LSRA

décalage logique à droite de A

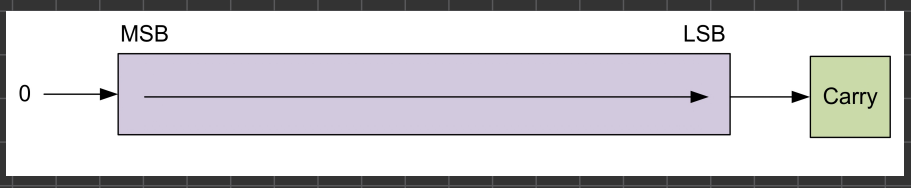
LSRB

décalage logique à droite de B

LSRD

décalage logique à droite de D

- décale chaque bit de 1 à droite
- remplis avec des 0 à gauche
- l'état du bit 0 est copié dans le drapeau carry
- décalage à droite est une division de $2^{\text{nb de décalage}}$ si c'est un nombre non-signé.
- l'état du carry nous renseigne si y'a un reste après la division
- Tous le modes d'adressages indexés et étendus



Décalage logique à gauche

LSL adresse

décalage logique à gauche d'un byte en mémoire

LSLA

décalage logique à gauche de A

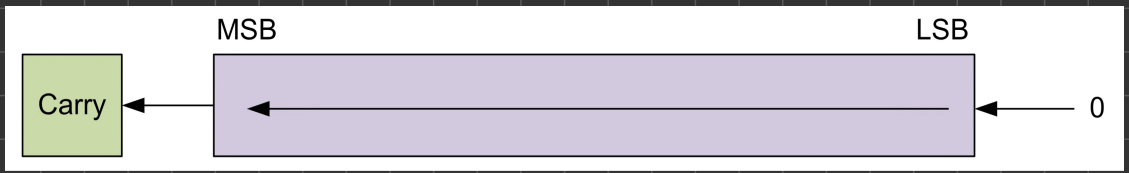
LSLB

décalage logique à gauche de B

LSLD

décalage logique à gauche de D

- décale chaque bit de 1 à gauche
- remplis avec des 0 à droite
- l'état du bit 7 est copié dans le drapeau carry
- multiplication de $2^{nb\ de\ décalage}$ si c'est un nombre non-signé
- l'état du carry nous renseigne si il y'a un reste à la division



Rotations

- La différence entre le décalage et la rotation, c'est que la rotation possède le bit de poids fort / faible (gauche / droite) dans le carry
- Utilisées afin d'implémenter des fonctions arithmétiques à précision étendue ou pour les déplacements multi bytes.
- Modifient le CCR selon le résultat de l'opération
- Les modes d'adressages étendus et tous les indexés fonctionnent

Rotation à gauche

ROL adresse

rotation à gauche d'un byte en mémoire

ROLA

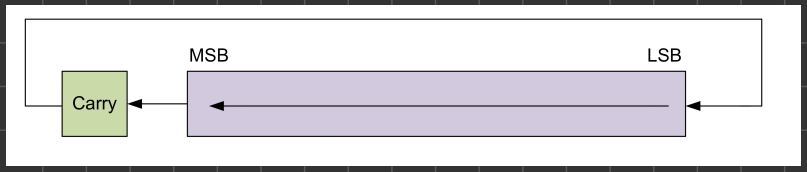
rotation à gauche de A

ROLB

rotation à gauche de B

ROLD

rotation à gauche de D



Rotation à droite

ROR adresse

rotation à droite d'un byte en mémoire

RORA

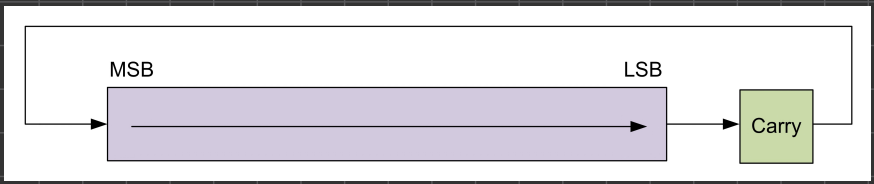
rotation à droite de A

RORB

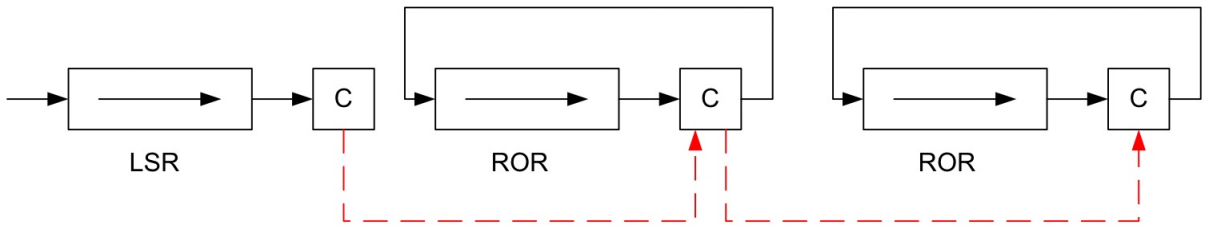
rotation à droite de B

RORD

rotation à droite de D



Exemple de décalage à droite d'une valeur sur 24 bits




```
; Division par 2 du nb de 24 bits avec les décalages et rotations
movb #$55,SH1      ;Initialise le MSB du nb de 24 bits
movb #$55,SH2      ;Initialise le byte médian du nb de 24 bits
movb #$55,SH3      ;Initialise le LSB du nb de 24 bits
lsr SH1            ;Shift à droite de 1 bits, le bits sortant se retrouve
                   ;dans le carry (il n'est pas perdu)
ror SH2            ;Rotation à droite de 1, le carry du shift précédent
                   ;se retrouve au bit de poids fort du byte médian, le
                   ;bit de poids faible du byte médian se retrouve dans
                   ;le carry
ror SH3            ;Rotation à droite de 1 du LSB. Le bit sortant du byte
                   ;médian se retrouve au bit 7 du LSB (il n'est pas
                   ; perdu), le carry le bit sortant du LSB, son état
                   ; indique si la division est entière ou pas
```

AND (ET logique, bit à bit)

ANDA adresse

ET logique bit à bit entre A et un byte en mémoire
résultat dans A

ANDB adresse

ET logique bit à bit entre B et un byte en mémoire
résultat dans B

EOR (OU EXCLUSIF, bit à bit)

EORA adresse

OU EXCLUSIF logique bit à bit entre A et un byte en mémoire
résultat dans A

EORB adresse

OU EXCLUSIF logique bit à bit entre B et un byte en mémoire
résultat dans B

OR (OU, bit à bit)

ORAA adresse

ou logique bit à bit entre A et un byte en mémoire
résultat dans A

ORBB adresse

ou logique bit à bit entre B et un byte en mémoire
résultat dans B

- AND, OR et EOR permettent les modes d'adressage immédiat, direct, étendu et tous les indexés
- Lorsque l'immédiat est utilisé, l'opérande est appelé "masque"
- AND, OR et EOR modifient les flags N et Z et mettent à 0 le flag V
- EOR est utile pour inverser les bits
- OR est utile pour mettre des bits à 1
- AND est utile pour mettre des bits à 0

```
anda #~4        ;Met à 0 le bit 2 de A
eora #$80       ;Inverse le bit de poids fort de A
orab loc1       ;Ou logique entre B et le contenu de loc1, les bits à 1 de loc1
                ;mettent à 1 les bits de B
```

Bit Set (BSET = OR)

Bit Clear (BCLR = AND)

BSET adresse mask

OU logique bit à bit entre valeur immédiate (mask) et le contenu du byte en mémoire

BCLR adresse mask

ET logique bit à bit entre valeur immédiate (mask) et le contenu du byte en mémoire

```
bset $810,#$4     ;Set bit number 2 in memory location $810
bclr $812,#$81    ;Clear bits 0 and 7 in memory location $812
```