Résumé Assembleur

Labo 3

08.11.2020

Owen Gombas
ISC1c

bra Entry

Modes d'adressage

- Il y'a différents modes d'adressage qui permettent d'atteindre la ROM, la RAM ou les registres
- Il y'a 3 groupes d'adressages:
  - Les opérandes se trouve dans les registres CPU: Pas besoin d'acceder à la ROM ou la RAM
  - Les opérandes se trouvent en mémoire
    1. On utilise directement leurs adresses absolues
    2. On utilise une adresse de référence à laquelle on ajoute/soustrait un offset afin d'accéder de manière indirecte à ces opérandes

Adressage inhérent INH

- Tous les opérandes se trouvent dans les registres CPU
- Pas d'accès à la mémoire car les données se trouvent dans le CPU
- Il est utilisé par les instructions qui agissent seulement sur les registres du CPU (A, B, D, X, Y, SP, PC, CCR)

Exemples:

ABA	$A + B \rightarrow A$
INX	$X + 1 \rightarrow X$
ASRA	$A \gg 1 \rightarrow A$

Adressage immédiat IMM

- Lorsque l'opérande est une constante
- l'opérande suit directement l'instruction
- La données est fixe, elle est inscrite dans le code (ROM). Elle peut donc seulement être modifiée en réassamblant le programme.
- Une constante est indiquée par #

Exemples

LDAA	#64	64 -> A
LDAA	#$64	$64 -> A
ADDA	#10	A+10 -> A
LDD	#$ffff	$ffff -> D

Adressage direct ou étendu DIR, EXT

- Quand l'opérande est une adresse constante de 8 bits (direct) ou 16 bits (étendu)
- L'adresse est indiquée dans le code soit par une valeur numérique ou un symbole (DS, DC, EQU)
- L'adressage direct permet d'accéder à la zone [0x00; 0xFF] (8 bits -> $2^8 = 255$) du memory map
- L'adressage étendu permet d'accéder à la zone [0x0000; 0xFFFF] (16 bits -> $2^{16} = 65535$) du memory map
- Il n'y pas le caractère # devant l'opérande

Exemples

```
LDAA   $64     contenu 0x64 -> A
STAB   255     contenu de B -> adresse 255
ADDA   $12     A + [$12] -> A
LDD    $8000   [$8000] -> D
```

Adressage Relatif REL

- Utilisé par les branchement et les sauts
- Un branchement est un saut, par exemple BRA Loop permet de sauté au label Loop (permet de créer des boucles)
- Un saut est une modification du compteur ordinal (PC). Après BRA Loop la valeur du PC sera celle du label Loop

Adressages indexés IDX

- Quand on utilise une adresse de base à laquelle on ajoute un offset
- Cette adresse de base se trouve dans le registre X, Y, SP ou PC.
- L'offset est une constante de 5bits, 9bits ou 16 bits
  ou le contenu d'un accumulateur
- On peut faire l'analogie avec les tableaux en langage haut niveau. (C, Java, C#, ...)
  On stock une adresse de base et on fait varier (increiment, decreiment) un offset.

LD... offset, registre d'index

registre d'index + offset -> destination

Avec un offset constant IDX1, IDX2

- l'offset est constant et codé sur 8 bits ou 16 bits.
  Il est connu lors de l'écriture du logiciel
- Elle est donc stockée en ROM

Exemples :

LDAA	0, X	contenu de l'adresse X+0 -> A	(0 offset 1 byte)
LDAA	64, X	contenu de l'adresse X+64 -> A	(offset 2 bytes)
STAA	-1, SP	A -> à l'adresse de (SP-1)	(offset 1 byte)
LDAA	5000, PC	contenu de l'adresse (PC+5000) -> A	

En C

```
char   Tab[10]
char   Var

Var = Tab[5]
```

En ASM

```
Tab : ds.b  10
Var : ds.b  1

ldx   #Tab
ldaa  5,x
staa  Var
```

Tab variable 10 bits
Var variable 1 bit

X ← contenu de Tab
A ← addr de Tab + 5
Var ← A

Avec un accumulateur comme offset

- L'offset est le contenu d'un accumulateur A, B ou D.
- Pour A et B l'offset est toujours positif
- Pour D l'offset peut être négatif
- L'offset est donc inconnu lors de l'écriture (variable)

Exemples

LDAA	B, X	contenu de l'adresse X+B -> A
LDAB	A, Y	contenu de l'adresse Y+A -> A
LDY	D, X	contenu de l'adresse X+D -> Y

Post / pré incrément / décrément (modification des registres d'index)

- Le but est de modifier le registre d'index
- On modifie soit avant ou après l'exécution de l'instruction.
  Cela dépend du placement de signe
  (incrément ou décrément entre 1 et 8)

LDAA Offset, ±X

incrément / décrément executé avant l'instruction LDAA

X ± offset -> X

X -> A

LDAA Offset, X±

incrément / décrément executé après l'instruction LDAA

X -> A

X ± offset -> X

Adressage indirects indexés avec offset constante 16 bits ou accumulateur D

[IDX2], [D,IDX]

- Semblable à l'adresse indexé classique
- On calcul donc une adresse avec le contenu d'un registre d'index (X, Y, SP ou PC) auquel on ajoute un offset (D ou constante 16 bits)
- Le calcul de l'offset ne donne pas directement le résultat (d'où adressage indirect) mais il retourne une nouvelle adresse contenant le résultat qui nous intéresse

```
LDS   [D,X]
```

Résumé des modes d'adressage

Mode d'adressage	Format	Abréviation	Description
Inhérent	INST (no externally supplied operands)	INH	Operands (if any) are in CPU registers
Immédiat	INST #opr8i or INST #opr16i	IMM	Operand is included in instruction stream 8- or 16-bit size implied by context
Directe	INST opr8a	DIR	Operand is the lower 8 bits of an address in the range $0000–$00FF
Etendu	INST opr16a	EXT	Operand is a 16-bit address
Relatif	INST rel8 or INST rel16	REL	An 8-bit or 16-bit relative offset from the current pc is supplied in the instruction
Indexé avec offset de 5 bits	INST oprx5,xysp	IDX	5-bit signed constant offset from X, Y, SP, or PC
Indexé avec pré décrémentation automatique	INST oprx3,–xys	IDX	Auto pre-decrement x, y, or sp by 1 ~ 8
Indexé avec pré incrémentation automatique	INST oprx3,+xys	IDX	Auto pre-increment x, y, or sp by 1 ~ 8
Indexé avec post décrémentation automatique	INST oprx3,xys–	IDX	Auto postdécrément x, y, or sp by 1 ~ 8
Indexé avec post incrémentation automatique	INST oprx3,xys+	IDX	Auto postinclémente x, y, or sp by 1 ~ 8
Indexé avec accumulateur comme offset	INST abd,xysp	IDX	Indexed with 8-bit (A or B) or 16-bit (D) accumulator offset from X, Y, SP, or PC
Indexé avec offset de 9 bits	INST oprx9,xysp	IDX1	9-bit signed constant offset from X, Y, SP, or PC (lower 8 bits of offset in one extension byte)
Indexé avec offset de 16 bits	INST oprx16,xysp	IDX2	16-bit constant offset from X, Y, SP, or PC (16-bit offset in two extension bytes)
Indexé indirect avec offset 16 bits	INST [oprx16,xysp]	[IDX2]	Pointer to operand is found at... 16-bit constant offset from X, Y, SP, or PC (16-bit offset in two extension bytes)
Indexé indirect avec un accumulateur comme offset	INST [D,xysp]	[D,IDX]	Pointer to operand is found at... X, Y, SP, or PC plus the value in D

abc	→ A or B or CCR
abcdxys	→A or B or CCR or D or X or Y or SP.
abd	→A or B or D
abdxys	→A or B or D or X or Y or SP
dxys	→D or X or Y or SP
msk8	→8-bit mask, some assemblers require # symbol before value
opr8i	→8-bit immediate value
opr16i	→16-bit immediate value
opr8a	→8-bit address used with direct address mode
opr16a	→16-bit address value
oprx0_xysp	→Indexed addressing postbyte code: oprx3,–xys Predecrement X or Y or SP by 1 . . . 8 oprx3,+xys Preincrement X or Y or SP by 1 . . . 8 oprx3,xys– Postdecrement X or Y or SP by 1 . . . 8 oprx3,xys+ Postincrement X or Y or SP by 1 . . . 8 oprx5,xysp 5-bit constant offset from X or Y or SP or PC abd,xysp Accumulator A or B or D, offset from X or Y or SP or PC
oprx3	→Any positive integer 1 . . . 8 for pre/post increment/decrement
oprx5	→Any integer in the range –16 . . . +15
oprx9	→Any integer in the range –256 . . . +255
oprx16	→Any integer in the range –32,768 . . . 65,535
page	→8-bit value for PPAGE, some assemblers require # symbol before this value
rel8	→Label of branch destination within –256 to +255 locations
rel9	→Label of branch destination within –512 to +511 locations
rel16	→Any label within 64K memory space trapnum — Any 8-bit integer in the range $30-$39 or $40-$FF
xys	→X or Y or SP
xysp	→X or Y or SP or PC