Microprocessor

101 ⟶ Decimale

%101 ⟶ Binaire

$101 ⟶ hexa decimal.

Rque

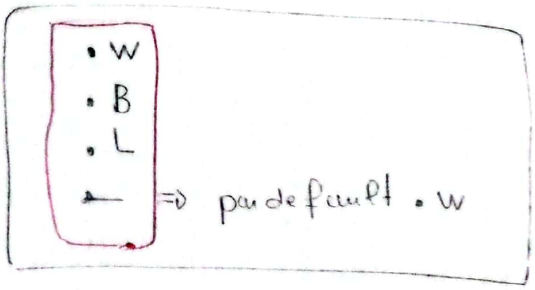
tjs on utilise l'hexadécimale

pour quoi on utilise l'hexadecimal.

$150 ⟶ 0001 0101 0000

0001 0101 0000

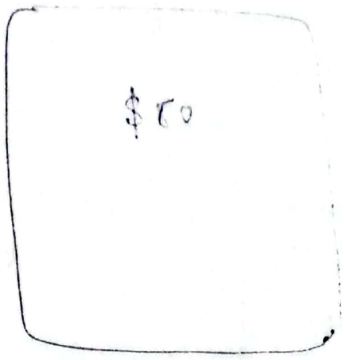
- Bit ≡ 0, 1
- Byte ≡ 1 octet ≡ 8 bits
- Word ≡ 16 bits
- Long Word ≡ 32 bits.



Memory 1 GB.

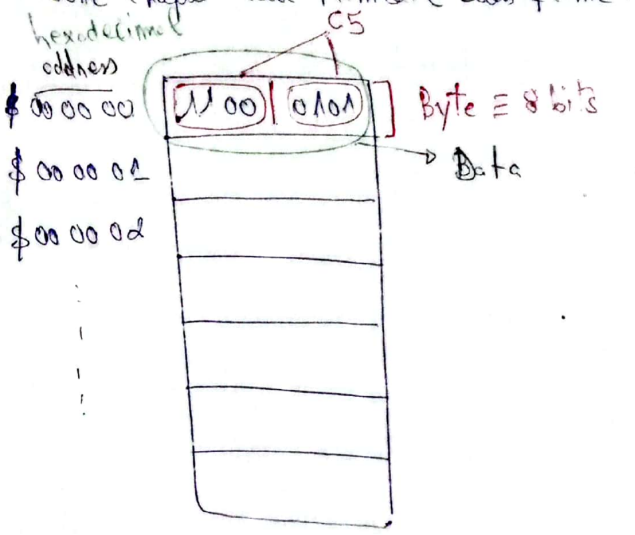
Data.

Address



$ 00 00 00 ⟶ $ FF FF FF

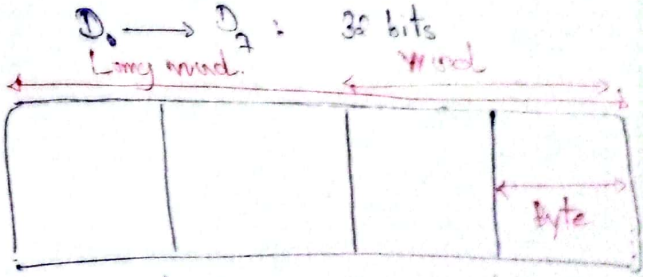
donc chaque case Mémoire sera forme :



Registre ≡ petite mémoire ≡ 24 bits par exemple

- Dans Dans le micro processor 2 type de registre

- Data registre :



68000 Microprocesseur

101 ⟶ Décimale

%101 ⟶ Binaire

$101 ⟶ hexa décimal.

Rque

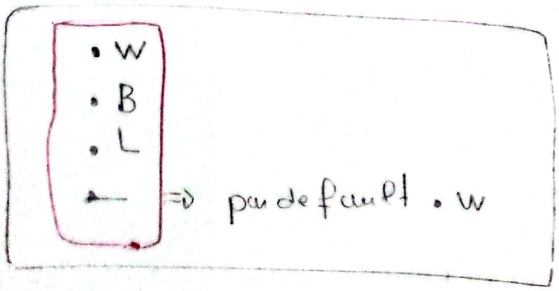
toujours on utilise l'hexadécimale

pour quoi on utilise l'hexadécimal.

$150 ⟶ 0001 0101 0000

0001 0101 000

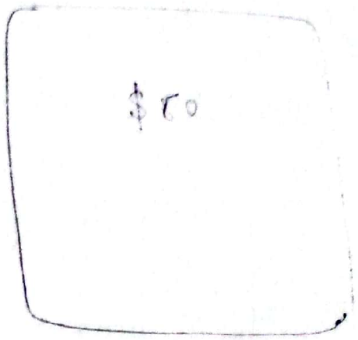
- Bit ≡ 0, 1
- Byte ≡ 1 octet ≡ 8 bits
- Word ≡ 16 bits
- Long Word ≡ 32 bits



Memory 1 GB.

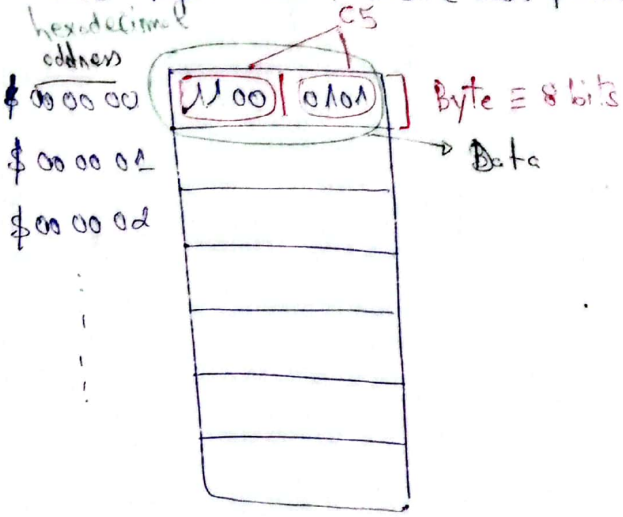
Data

Address



$? ? ? ? ? ? ⟶ $ FF FF FF

donc chaque case mémoire sera former.

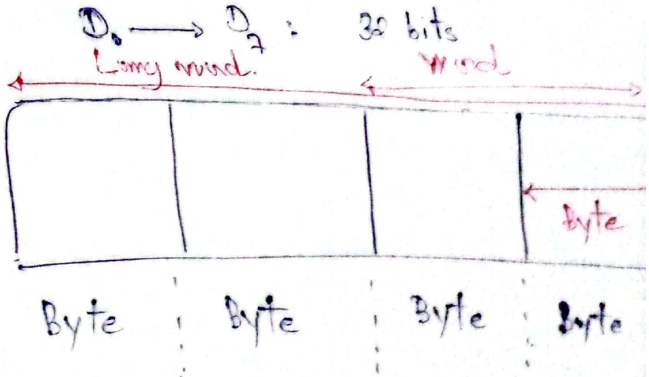


Registre ≡ petite mémoire ≡ 24 bit

par exemp

- Dans le microprocesseur 2 type de regi

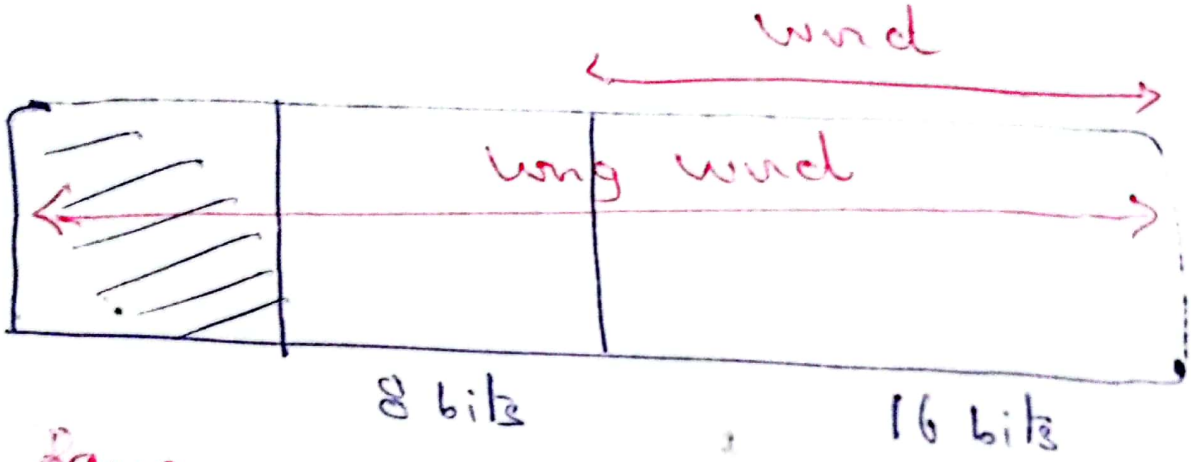
- Data registre :



- Registre d'adresses :

$A_0 \longrightarrow A_{15}$

24 bits



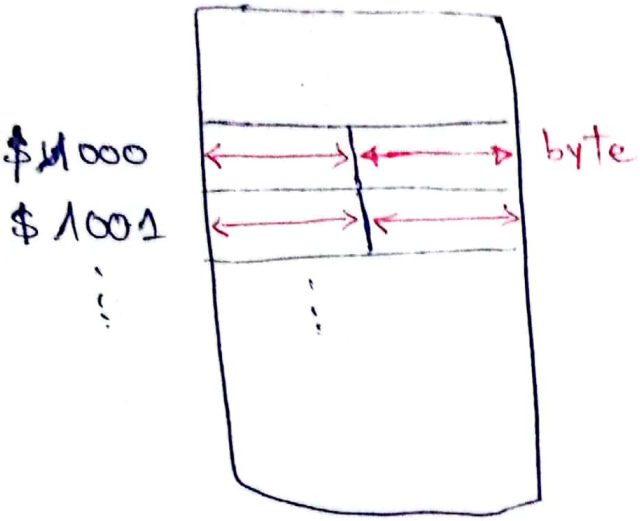
Rque

on travail seulement avec word ou long word, dans le registre d'adresses

Directive Instructions

ORG : origine.

ORG $1000 : on début par la case d'adresse 1000 :



la forme générale :

```
ORG $1000
    [ code
SIM HALT
    [ Data → DC : Définie Constant
           → Ds : Définie stockage.
END
```

Addressing Modes:

1/ Immediate

Move .B / .W / .L # 5, D_2

↳ décimal

: cost

= constante.

if fait nombre hexadecimale ($).

B = 05

W = 00 05

L = 00 00 00 05

Donc si D_1 = 33 33 33 33

⇒ Move.B #5, D_1

D_1 = 33 33 33 **05**

⇒ Move.W

D_1 = 33 33 00 05

⇒ Move.L

D_1 = 00 00 00 05

Exemples:

Move.L #140, D_5

1ère: on rembre 140 en Hexa.

$(140)_{10} = (8C)_{16}$

⇒ Move.L #$8C, D_5

⇒ D_5 = 00 00 00 8C

exp2:

- move.B #140, D_3

⇒ move.B #$8C, D_3

⇒ D_3 = -- -- -- 8C.

- move.W #140, D_3

⇒ D_3 = -- -- 00 8C

exp3:

- move.B #$A5, A_2.

Rque
tout opération se fait sur octets registre
soit .B ou .W seulement

⇒ retourne erreur

- move.W, #$A5, A_1

A_1 = [FF] 00 A5

Dans cette cas

A_1 = [FF] 00 00 A5.

Rque
- Dans le cas du .W
le registre d'octets vaut 00 ou FF
- Dans le cas du .L
reste comme elle est.

pour .w

word

positive

0000 → 7FFF

00

Négative

8000 → FFFF

FF

exemples :

Move.w #$A5, $A1

8000 < 00A5 < 7FFF

⇒ A_1 = [illegible] 00 00 A5.

exp :

move.w #-2, A_3

FFFE

⇒ A_3 = [illegible] FF FFFE

---

move.L #X+4, A_2

ORG $40E00

X : DS.L 1.

explication de ce code

si X = 5

5+4 = 9.

⇒ Move.L #9, A_2.

X = toujours est un adress donc dans

ce cas X = $40E00

X + 4 = $40E00 + 4

= $40E04

⇒ Move.L #$40E04, A_2

⇒ A_2 = [illegible] 04 0E 04

exp :

Move.B #-2, D_1

FE

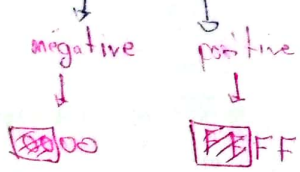
⇒ D_1 = [illegible] FE

Move.L $1000.L , D_4
.W .W

Remarque :

si .w : on le reparen comme l'adress registre .

si .w ⇒ - - 1000



explication

Move.L $1000.L , D_4

ici c'est à dire trouve la case d'adress 1000 et prend le contenue (.L) et mettre dans D_4 .

si

$1000	56
$1001	78
$1002	9A
$1003	BC

donc dans cette cas.

~~D_4 = BC 9A 78 56~~

D_4 = 56 78 9A BC

Move.B $F000.L , D_4

$F000	78

⇒ D_4 = - - - 78

2/° Register Direct :

Move.L D_5 , D_1

si D_5 = 11 11 11 11

D_1 = 00 00 00 00

⇒ D_1 = 11 11 11 11

et D_5 = 11 11 11 11

Move.W D_1 , A_5

D_1 = 00 00 A5 00

⇒ .W A5 00

⇒ A_5 = FF A5 00

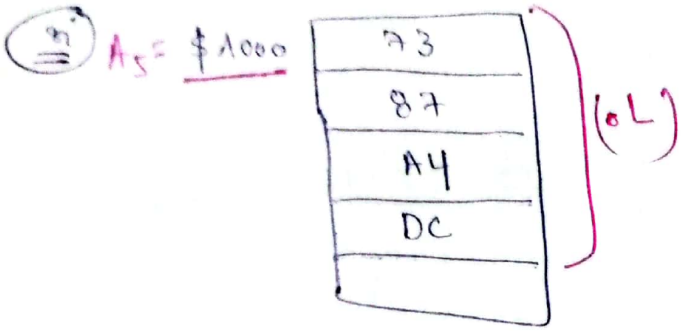
copier - coller

4° Registre Indirect

Move.L (A5), D1 tq A5 = $1000

explication : →

c-à-d : aller à A5 = $1000, emprunter le contenue de la case $1000 (.L) et le mettre dans D1



⇒ D1 = 7387A4DC

Reque :

Move.L (A5), D1
↓↓↓
$1000

c-à-d La seule difference entre indirect et absolut c'est $1000 et (A5).

tq A5 = $1000

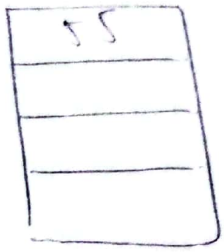
6° Registre indirect prédécrémenté

• par convention (A5)+
-(A5)

exp :

Move.B (A5)+1, D1

A5 = $001000 $1000



⇒

.B	+1	A5 = $1001
.W	+2	A5 = $1002
.L	+4	A5 = $1004

⇒ D1 = ---- ---- ---- 55

A5 = $001001

exp :

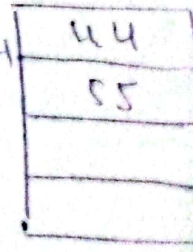
Move.W -(A5), D1

tq A5 = $001006.

.B	-1
.W	-2
.L	-4

A5 = $001004

donc allons à $1004 : $1004



⇒ A5 = $001004

D1 = ---- 4455

Exp :

Move . B D_0 , -(A_4)

D_0 = 4E CC 1F B3

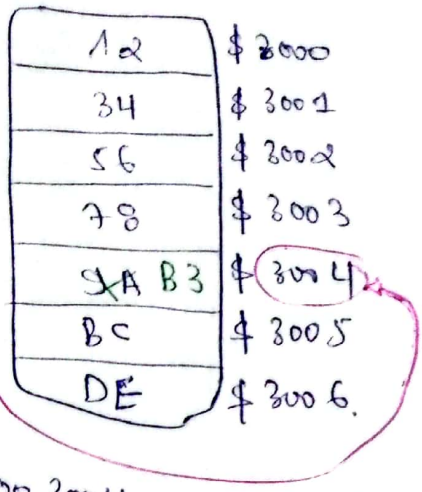
A_4 = 00 3005

⇒ A_4 = 00 3004

⇒

Move . B : on prend de D_0 juste (AB3).

-(A_4) et comme . B donc A_4 = 00 3004



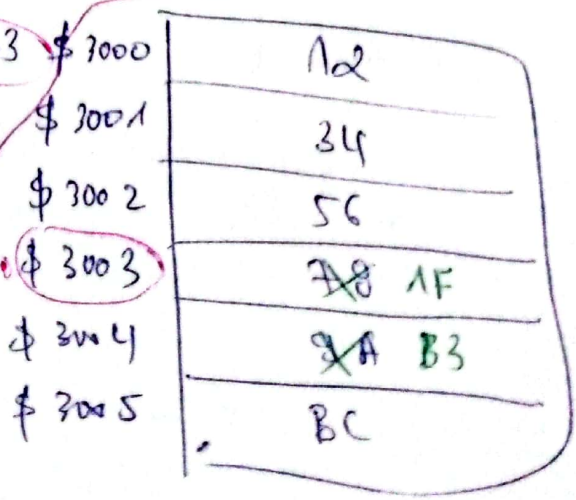
⇒ Alors si la case $ 00 3004 et on remplace le contenue de (00 3004) par (B3)

⇒

Move . W D_0 ,(-A_4)

A_4 = 00 3003

⇒ D_0 = 4E CC 1F B3



Move.w D_5, 40(A5) ⇒ A5 + 40

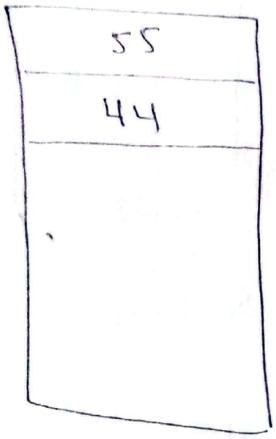
⇒ & A5 = $5000

~~⇒ 40(A5) = $5040~~

⇒ 40(A5) = $5040

D_5 = - - - - 5544

at A5 note $5000



$5040

exp:

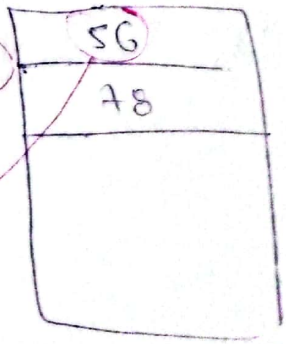
move.B -3(A5), D_5.

A5 = $1003.

⇒ -3(A5) = $1000

move.B

$1000



⇒ D_5 = - - - - - - 56

Move.L (d8) (A, X : W / L)

(-127 ÷ +127)

Data registre
ou
Adresse

exp :

Move.L -4 (A3, D2.W)

= A3 - 4 + D2.W

exp :

Move.L D5, -4 (A3, D2.W)

D5 = 00 11 22 33

A3 = 00 01 08 E0

D2 = 00 1F 90 F0

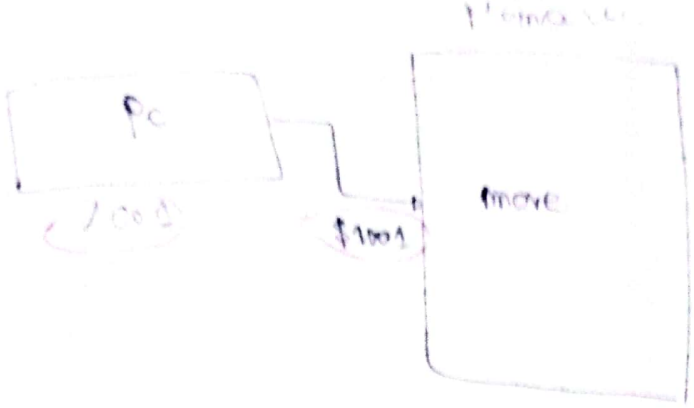
$00 01 08 E0 - 4 + $FF FF 90 F0

négatif

on convertit tout en décimale

67808 - 4 - 28432

= 39372 ≡ $00009 9CC



91° Programm counter Relative (position indépendant)

Move . L #$C24A1B00, X

Move . L X(PC), D_2

ORG $2000

X : DS.L 1

⇒ X = $2000

④

$2000	C2
$2001	4A
$2002	1B
$2003	00

④ move . L X , D_2 ⇒ move . L , $2000 , D_2

⇒ D_2 = C2 4A 1B 00

② Sauf (PC) ne s'agit pas (position indépendant)

→ reste comme ça.

- Arithmetic instructions :

ADD, ADDA, ADDQ, ADDI, ADDX.

Sub, SubA, SubQ, SUBI, SUBX;

NEG, NEGX

CLR

MULU, MULS (Unsigned, Signed)

DIVU, DIVS

ABCD (ADD BCD)

SBCD (Sub BCD)

CMP, CMPA, CMPI, CMPM.

une limite d'utilisation

ADD:

size = Byte, word, Long.

Source = Any adressing mode

Destination = Can't be adress register

- one of the source or Destination must be Data register.

Exp:

$D_2 = 10\,1AC\,01\,0$, $A_1 = 00\,00\,20\,00$

ADD.W $(A_1), D_2$

$D_2 = [D_2].w + M[A_1].w$

$= C0\,10 + C2\,40$

$\Rightarrow D_2 = 3250$

C2	$2000
40	$2001

Mémoire de A_1.

ADD.B A_1, D_2

$D_2 = [D_2].B + [A_1].B.$

$D_2 = 10 + 00 = 10.$

ADD.L A_1, D_2

$\Rightarrow D_2 = 10\,1A\,C0\,1D + 00\,00\,20\,00$

$\boxed{D_2 = 10\,1A\,E0\,1D}$

ADD.W D_2, A_2

↳ ennem, destination can't be adress register

ADDX comprise entre

ADDX	D_1, D_2
ADDX	$-(A_1), -(A_2)$

si $X = 1$

ADDX D_1, D_2

$\Rightarrow \boxed{X + D_1 + D_2 = D_2}$

sinon : on applique ADD normale.

exp :

ADD.B A_2, D_2

$D_2 = [D_2].B + [A_2].B.$

$D_2 = 10 + 00 = 10.$

ADD.L A_1, D_2

$\Rightarrow D_2 = 10\,AC\,01\,D + 00\,00\,80\,00$

$\boxed{D_2 = 10\,AE\,01\,D}$

ADD.W D_2, A_2

↳ ennem, destination can't be other register

ADDX

Comprise entre

ADDX	D_1, D_2
ADDX	$-(A_2), -(A_2)$

si $X = 1$

ADDX D_1, D_2

$\Rightarrow \boxed{X + D_1 + D_2 = D_2}$

Si mem : on applique ADD normale.

SUB : (Même les conditions que ADD, ADDA, ...)

SUB D_1, D_2 $\quad [D_2] = [D_2] - [D_1]$

SUBX D_1, D_2 $\quad [D_2] = [D_2] - [D_1] - [X]$

NEG :

- Size ≡ Byte, word, Long word
- Camlt le adress registres

Exp :

NEG.L D_0 $\quad$ tq $D_0 = 0123CA78$

FF FF FF FF
0123 CA 78 −
FE DC 35 87
1 ↑
FE DC 35 88 = D_0

NEG.W D_1 $\quad D_1 = 0043ABFE$

FF FF
CA 78
FFFF
ABFE −
5401
1 ↑
5402

⇒ $D_1 = 00435402$

NEGX.

si X = 1

NEGX.W D_1

5402 ⇒ D_2 = 0043 5401

(on soustracte X = 1)

CLR :

n'existe pas sur les Adress registers.

CLR.L D_0 ⇒ D_0 = 00 00 00 00

MULU : (Unsigned)

MULU D_1 (word), D_2 (word) ⇒ Long word

si D_1 = 5678 ABCD

D_2 = 1234 5678

⇒ (ABCD) × (5678)