

ARCHITECTURE DES ORDINATEURS

(8)- INTRODUCTION à l' **ASSEMBLEUR 'x86' (III)**

' TRAITEMENT CONDITIONNEL '

**Ou quels sont les 'outils architecturaux des CPU 'x86'
pour implémenter des ' TRAITEMENTS
CONDITIONNELS ' ?**

(8)- INTRODUCTION à l' ASSEMBLEUR 'x86' (**III**)

TRAITEMENT CONDITIONNEL

Définition : (rappel)

Tout TRAITEMENT (ou 'ensemble d'instructions') assujetti / asservi à l'état VRAI (ou FAUX) d'une PROPOSITION dite 'CONDITION'

NB:

≠ « STRUCTURE CONDITIONNELLE »

(8)- INTRODUCTION à l' ASSEMBLEUR 'x86' (**III**)

TRAITEMENT CONDITIONNEL

Définition : (rappel)

Tout TRAITEMENT (ou 'ensemble d'instructions') assujetti / asservi à l'état VRAI (ou FAUX) d'une PROPOSITION dite 'CONDITION'

- ❶ TRAITEMENT CONDITIONNEL => Anticipation dans le temps
- ❷ TRAITEMENT CONDITIONNEL → Au moins 1 fois dans tout prg !

(8)- INTRODUCTION à l'ASSEMBLEUR 'x86' (*III*)

TRAITEMENT CONDITIONNEL

Corollaire 1 :

Toute CONDITION peut toujours être ramenée à l'expression d'une **égalité** / inégalité.

ALGO:

IF (op1=op2) THEN

<Inst1>

<Inst2>

...

ELSE

...

ENDIF



C:

IF (op1==op2)

{

<Inst1>

<Inst2>

...

}

ELSE {...}

(8)- INTRODUCTION à l'ASSEMBLEUR 'x86' (*III*)

TRAITEMENT CONDITIONNEL

Corollaire 1 :

Toute CONDITION peut toujours être ramenée à l'expression d'une **égalité** / inégalité.

<u>ALGO:</u>	<u>'C':</u>	<u>'ASM x86':</u>
IF (op1=op2) THEN <Inst1> <Inst2> ... ELSE ... ENDIF	IF (op1==op2) { <Inst1> <Inst2> ... } ELSE {...}	CMP op1,op2 JE Egal JMP Autre Egal: Inst1 Inst2 ... Autre : <Suite>

(8)- INTRODUCTION à l'ASSEMBLEUR 'x86' (*III*)

TRAITEMENT CONDITIONNEL

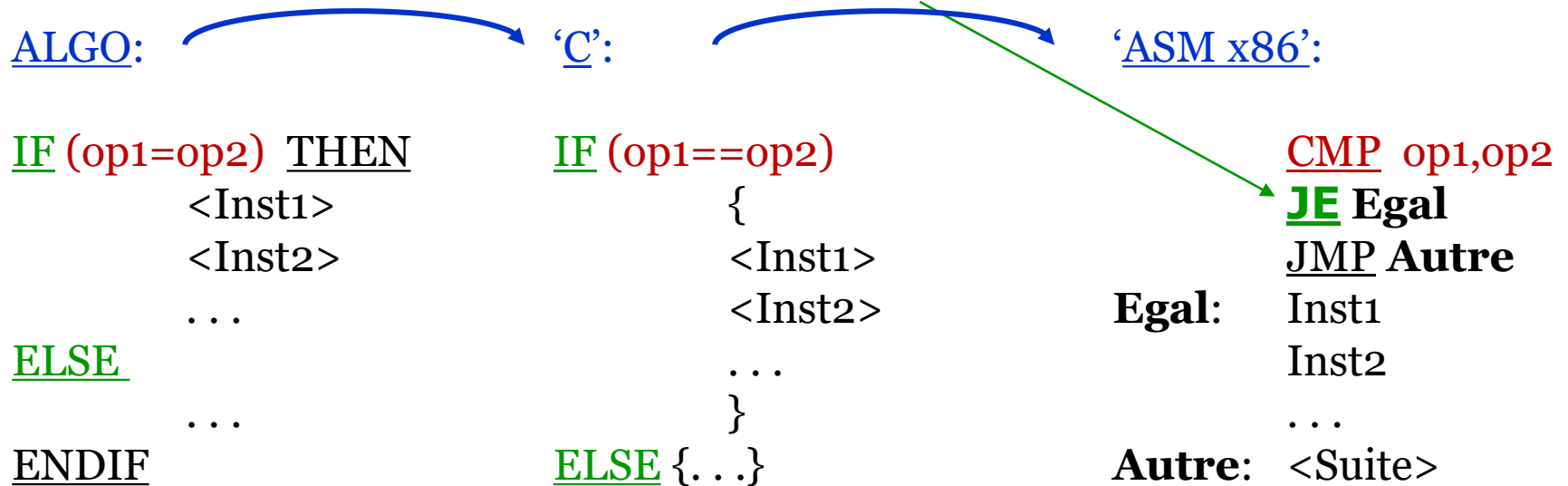
L'expression de l'EGALITE est basiquement une COMPARAISON

<u>ALGO:</u>	<u>'C':</u>	<u>'ASM x86':</u>
<u>IF</u> (op1=op2) <u>THEN</u> <Inst1> <Inst2> ... <u>ELSE</u> ... <u>ENDIF</u>	<u>IF</u> (op1==op2) { <Inst1> <Inst2> ... } <u>ELSE</u> {...}	<u>CMP</u> op1,op2 <u>JE</u> Egal <u>JMP</u> Autre Egal: Inst1 Inst2 ... Autre: <Suite>

(8)- INTRODUCTION à l'ASSEMBLEUR 'x86' (*III*)

TRAITEMENT CONDITIONNEL

JE: **J**ump if **E**qual
(choix d'un traitement
[conditionné par le résultat
de CMP] débutant à
l'étiquette 'EGAL')



(8)- INTRODUCTION à l'ASSEMBLEUR 'x86' (*III*)

TRAITEMENT CONDITIONNEL

Q: comment « repérer » l'ETAT
« EGALITE » ? Quel OUTIL (matériel)
mettre en place pour repérer cet état ?

'C':

IF (op1==op2)
{
 <Inst1>
 <Inst2>
 ...
}
ELSE {...}

'ASM x86':

CMP op1,op2
JE Egal
JMP Autre
Egal: Inst1
 Inst2
 ...
Autre: <Suite>

(8)- INTRODUCTION à l'ASSEMBLEUR 'x86' (*III*)

TRAITEMENT CONDITIONNEL

JE: **J**ump if **E**qual
(choix d'un traitement
[conditionné par le résultat
de CMP] débutant à
l'étiquette 'EGAL')



L'état '**EQUAL**' est décisif pour l'exécution
de la séquence <Inst1, ...>: il dépend lui
même de **l'état** généré par '**CMP**', **ETAT**
inscrit dans le **registre (IF)** (à venir)

ALGO:

'C':

'ASM x86':

```
IF (op1=op2) THEN  
    <Inst1>  
    <Inst2>  
    ...  
ELSE  
    ...  
ENDIF
```

```
IF (op1==op2)  
    {  
        <Inst1>  
        <Inst2>  
        ...  
    }  
ELSE { ... }
```

```
CMP op1,op2  
JE Egal  
JMP Autre  
Egal:  Inst1  
       Inst2  
       ...  
Autre: <Suite>
```

(8)- INTRODUCTION à l' ASSEMBLEUR 'x86' (*III*)

TRAITEMENT CONDITIONNEL

Corollaire 2:

Toute INSTRUCTION génère un 'résultat principal' (AIM), indexé d'éventuelles informations (Etats/ Résultats) secondaires.

Exemple 1:

Si AX= 00FFh

ADD AX, 1

=>

AX = 0100h

& déb (LSB -> MSB)

Exemple 2:

Si AX= FFFFh

ADD AX, 1

=>

AX = 0000h

& déb (LSB -> MSB)

& déb (MSB -> 'retenue')

(8)- INTRODUCTION à l' ASSEMBLEUR 'x86' (*III*)

TRAITEMENT CONDITIONNEL

Corollaire 2:

Toute INSTRUCTION génère un 'résultat principal' (AIM), indexé d'éventuels informations (Etats/ Résultats) secondaires.

Exemple 1:

Si AX= 00FFh

ADD AX, 1

=>

(1) AX = 0100h

(2) déb (LSB -> MSB)

Exemple 2:

Si AX= FFFFh

ADD AX, 1

=>

(1) AX = 0100h

(2) déb (LSB -> MSB)

(2) déb (MSB -> 'retenue')

Ces (Etats/ States) secondaires sont '**immédiatement**' induits

(8)- INTRODUCTION à l'ASSEMBLEUR 'x86' (*III*)

TRAITEMENT CONDITIONNEL

Exemple: Fiches techniques « ADD », « SUB ».

→ Résultats Primaires & Secondaires

Résultats Iaires

AND AX, 1 $AX \leftarrow AX \text{ AND } 1$

SUB AX, 1 $AX \leftarrow AX - 1$

Résultats IIaires « Immédiats »

O	S	Z	A	P	C
0	x	x	i	x	0
x	x	x	x	x	x

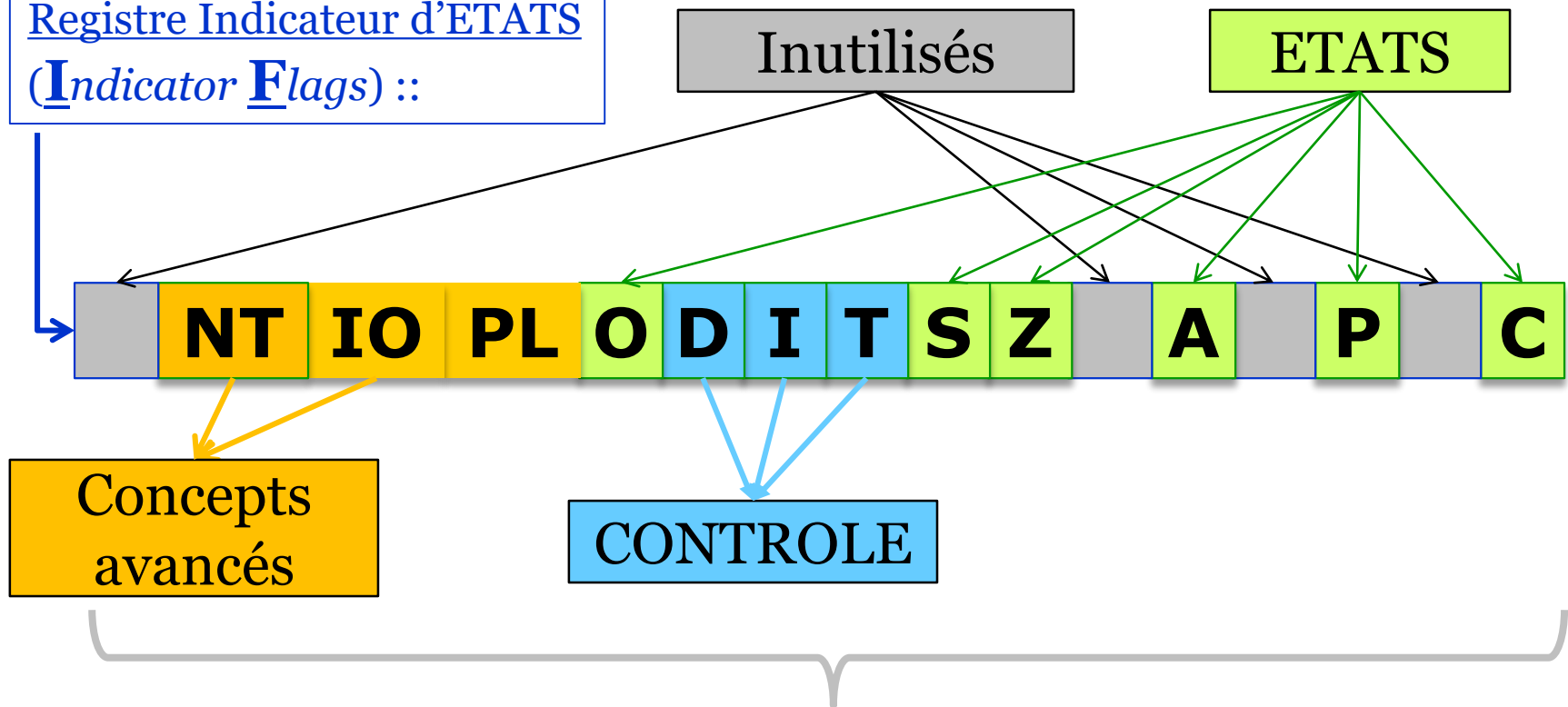
Etats → 'o' : mis à 0 ; 'x' : affecté ; 'i' : indéfini ; '1' : mis à 1

(8)- INTRODUCTION à l' ASSEMBLEUR 'x86' (*III*)

TRAITEMENT CONDITIONNEL :

Registre Indicateur d'ETATS '80286'

Registre Indicateur d'ETATS
(Indicator Flags) ::



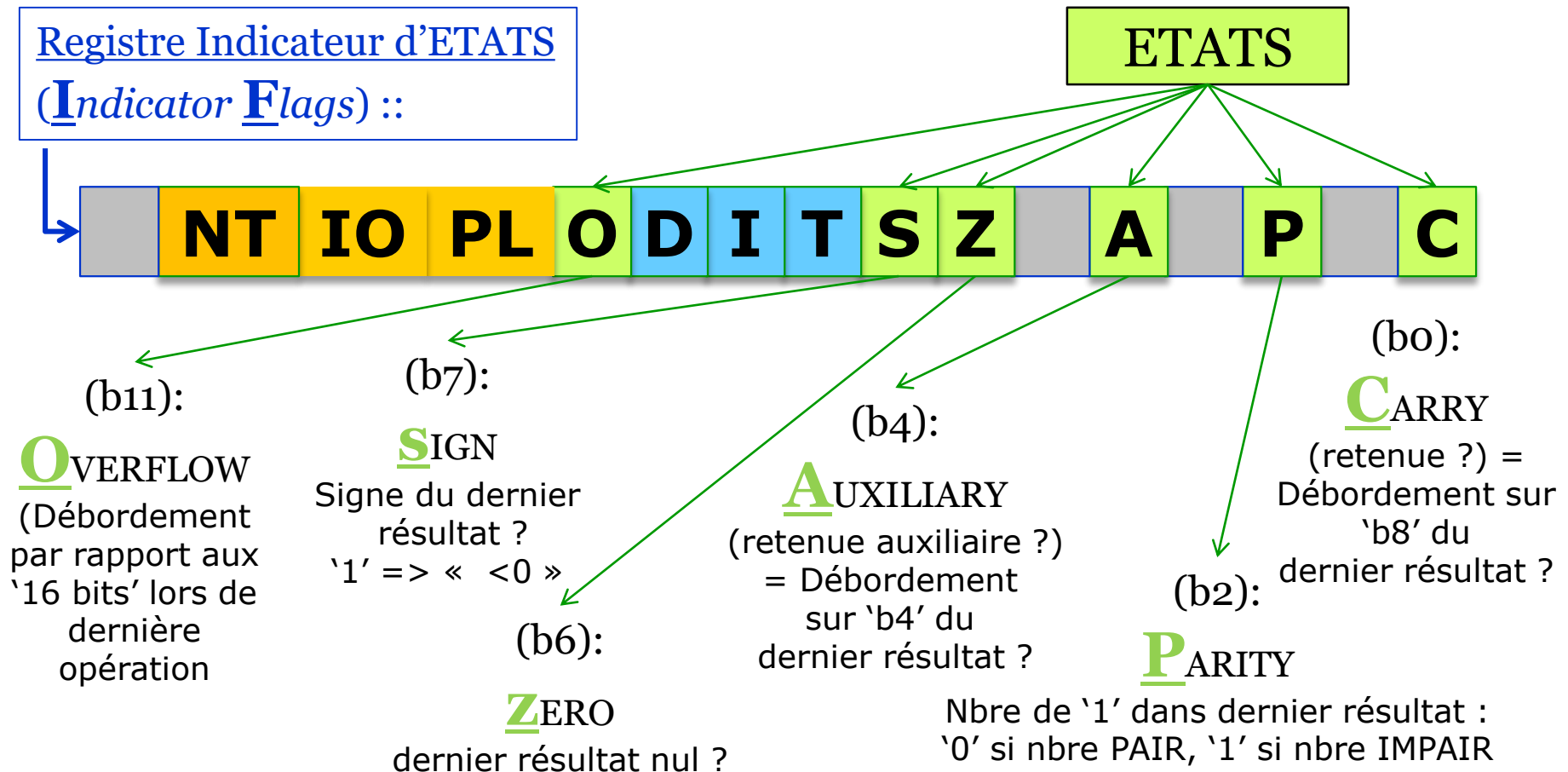
Registre '16 bits', tous bits accessibles **séparément**

(8)- INTRODUCTION à l'ASSEMBLEUR 'x86' (III)

TRAITEMENT CONDITIONNEL :

Registre Indicateur d'ETATS '80286'

Registre Indicateur d'ETATS
(Indicator Flags) ::



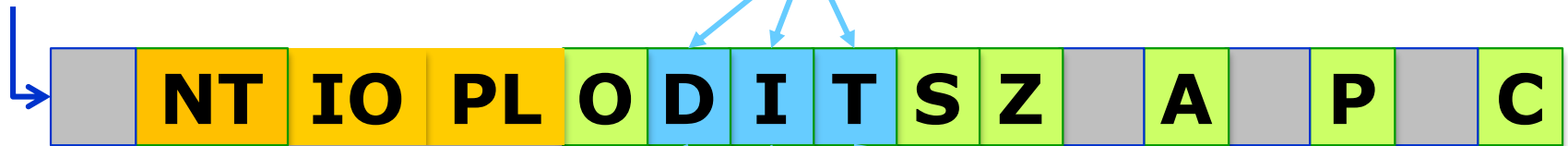
(8)- INTRODUCTION à l'ASSEMBLEUR 'x86' (III)

TRAITEMENT CONDITIONNEL :

Registre Indicateur d'ETATS '80286'

Registre Indicateur d'ETATS
(Indicator Flags) ::

CONTROLE



(b10):

DIRECTION (Up / Down) :

Choix du Sens de comptage
croissant (0)/ décroissant (1)

Illustr: « MOVS », « CMPS » ..

(b9):

INTERRUPT

Choix de l'autorisation (1) ou
interdiction (0) des
interruptions masquables

(b8):

TRAP

Choix de l'autorisation (1)
ou interdiction (0)
des interruptions générant
le mode 'PAS à PAS'

(8)- INTRODUCTION à l' ASSEMBLEUR 'x86' (*III*)

TRAITEMENT CONDITIONNEL :

Registre Indicateur d'ETATS '80286'

Registre Indicateur d'ETATS
(Indicator Flags) ::

Concepts
avancés



(b14):

NESTED TASK

Indique (NT=1) qu'une
tache a invoqué une autre
via l'instruction 'CALL')

(b13-b12):

I/O PRIVILEGE LLEVEL

Niveau de PL requis
pour une procédure
faisant appel aux
instruction E / S :: IN, OUT, INS ...

(8)- INTRODUCTION à l' ASSEMBLEUR 'x86' (**III**)

TRAITEMENT CONDITIONNEL

Registre Indicateur d'ETATS :: (Indicator Flags)

Exemple d'application

*; Analyse (**P**aire/**I**mpaire) de la donnée logée à l'@ '1000h'*

MOV AX, [1000H]

MOV DX, AX

AND DX, 1 ; (3)

SUB DX, 1 ; (4)

[JZ TRAITE_IMPAIR]

JMP TRAITE_PAIR

<Seq_Instr1>

< Seq_Instr2>

TRAITE_IMPAIR:

TRAITE_PAIR:

Traitement Nbres. IMPAIRS

Traitement Nbres. PAIRS_NUL

Branchement à 'TRAITE_IMPAIR' si (dernière/précédente) instruction 'SUB DX,1' a généré un résultat nul (équivalent à b0(DX)=1)



Les instructions 3 & 4 engendrent une équivalence entre la valeur du bit b0(DX) (désignant la parité de DX) et le bit 'Z' du registre (IF): on tire profit de cette équivalence pour notre traitement conditionnel

NB: avec 'JZ', la condition est **EXPLICITEMENT** rapportée au bit 'Z' du registre 'IF'

(8)- INTRODUCTION à l' ASSEMBLEUR 'x86' (III)

TRAITEMENT CONDITIONNEL

Instructions de 'SAUT CONDITIONNEL' : (JCond) :: Classement par TYPE

GENERAL

- Z** → Jump if **Z**ero / Jump if **N**ot **Z**ero (JZ / JNZ)
- C** → Jump if **C**arry / Jump if **N**ot **C**arry (JC / JNC)
- P** → Jump if **P**arity / Jump if **N**ot **P**arity (JP / JNP)
- Jump if **E**qual / Jump if **N**ot **E**qual (JE / JNE)
- Jump if **CX=Z**ero (JCXZ)

Nbres SIGNÉS

- Jump if **G**reater / Jump if **N**ot **G**reater (JG / JNG)
- Jump if **G**reater or **E**qual / Jump if **N**ot **G**reater or **E**qual (JGE / JNGE)
- S** → Jump if **S**ign / Jump if **N**ot **S**ign (JS / JNS)
- O** → Jump if **O**verflow / Jump if **N**ot **O**verflow (JO / JNO)

Usage
EXPLICITE
du registre
'IF'
→ Sol **HARD**

Usage
IMPLICITE
du registre
'IF'
→ Sol **SOFT**

(8)- INTRODUCTION à l'ASSEMBLEUR 'x86' (*III*)

TRAITEMENT CONDITIONNEL

Instructions de 'SAUT CONDITIONNEL' : (JCond) :: Classement par TYPE (SUITE)

Nbres NON-SIGNÉS

- ☐ → **J**ump if **A**bove / **J**ump if **N**ot **A**bove (JA / JNA)
- ☐ → **J**ump if **A**bove or **E**qual/ **J**ump if **N**ot **A**bove or **E**qual (JAE / JNAE)
- ☐ → **J**ump if **B**elow / **J**ump if **N**ot **B**elow (JB / JNB)

Usage
IMPLICITE
du registre
'IF'
→ Sol **SOFT**

Attention: liste des 'jump' conditionnels non complète

(8)- INTRODUCTION à l' ASSEMBLEUR 'x86' (**III**)

TRAITEMENT CONDITIONNEL

Instructions de 'SAUT CONDITIONNEL' :: Exemples d'application

Exemple 1:

*; Recherche du plus petit des nombres **NON-SIGNÉS***

; small est un identifiant qcq

	MOV small, AL	<i>; on suppose AL : le + petit</i>
	CMP small, BL	<i>; .. On compare avec BL</i>
	<u>JBE</u> L1	<i>; saut à 'L1' si AL = small '<u>antérieur</u>' BL</i>
	MOV small, BL	<i>; sinon, BL devient small</i>
L1 :	CMP small, CL	<i>; . . . puis compare avec CL</i>
	<u>JBE</u> L2	<i>; . . . etc</i>
	MOV small, CL	
L2 :	<suite>	<i>; suite avec CL plus petit des 3 nbres</i>
		<i>; . . . <u>non-signés</u></i>

(8)- INTRODUCTION à l' ASSEMBLEUR 'x86' (**III**)

TRAITEMENT CONDITIONNEL

Instructions de 'SAUT CONDITIONNEL' :: Exemples d'application

Exemple 2:

*; Recherche du plus petit des nombres **SIGNÉS**
; small est un identifiant qcq*

	MOV small, AL	<i>; on suppose AL : le + petit</i>
	CMP small, BL	<i>; .. On compare avec BL</i>
	<u>JNG</u> L1	<i>; saut à 'L1' si AL = small '<u>+ petit</u>' que BL</i>
	MOV small, BL	<i>; sinon, BL devient small</i>
L1 :	CMP small, CL	<i>; . . . puis compare avec CL</i>
	<u>JNG</u> L2	<i>; . . . etc</i>
	MOV small, CL	
L2 :	<suite>	<i>; suite avec CL plus petit des 3 nbres</i>
		<i>; . . . <u>signés</u></i>

(8)- INTRODUCTION à l'ASSEMBLEUR 'x86' (*III*)

TRAITEMENT CONDITIONNEL

Instructions de manipulation des bits de CONTRÔLE

BITS de CONTRÔLE

D → **SeT** **D**irection Flag / **CLear** **D**irection Flag (**STD** / **CLD**)

I → **SeT** **I**nterrupt Flag / **CLear** **I**nterrupt Flag (**STI** / **CLI**)

BITS d'ETATS

C → { **SeT** **C**arry Flag / **CLear** **C**arry Flag (**STC** / **CLC**)
CoMplement **C**arry Flag (**CMC**)

REGISTRE d'ETATS

- Load **AH** with **Flag** / Store **AH** to **Flag** (**LAHF** / **SAHF**)
- **PUSH** **Flag** / **POP** **Flag** (**PUSHF** / **POPF**)

(8)- INTRODUCTION à l' ASSEMBLEUR 'x86' (**III**)

TRAITEMENT CONDITIONNEL

Instructions de manipulation des bits de CONTROLE

RESUME . . .