Tpn°2

Les instructions

Les instructions arithmétiques

Instruction ***MOV***

est l'instruction la plus utilisée en assembleur.

- But : copie le contenu de l'opérande source dans l'opérande de destination.
- Syntaxe :

  MOV destination, source

- Flags :

  Aucun flag n'est modifié par cette instruction.

- Exemple:

```
MOV AX, 34     (AX <- 34)
MOV DX, CX     (DX <- CX)
```

Les instructions arithmétiques

Instruction ***ADD***

- But : Ajouter le contenu de l'opérande source au contenu de l'opérande de destination
- Syntaxe :

  ADD destination, source
- Flags :

  ZERO (z) et **CARRY** (c) **(et overflow** (o)**)** modifiés selon le résultat.
- Exemples :

  ADD AX, 14 (AX<- AX+14)

  ADD DX, CX (DX <- DX + CX)

  ADD AX, 97 (capacité : AX <- 1000h et CARRY = 1)

Les instructions arithmétiques

Instruction ***SUB***

- But :Soustraire le contenu de l'opérande source au contenu de l'opérande de destination
- Syntaxe :

  SUB destination, source
- Flags :

  ZERO et **SIGNE** modifiés selon le résultat.
- **Exemples** :

  SUB AX, 4 (AX<- AX - 4)

  SUB CX, 1 (CX <- CX -1)

Les instructions arithmétiques

Instruction **MUL:**

- But: multiplier le contenu de l'opérande source au contenu du registre **AX** et placer le résultat dans le couple **DX-AX**
- Syntaxe :

  MUL source

  (rem : *source* ne peut pas être une valeur immédiate!)
- Flags :

  OVERFLOW, CARRY = 0 si DX = 0

  ZERO, SIGNE modifiés mais indéfinis
- Exemple :

  MUL CX (DX-AX <- AX * CX)

Les instructions arithmétiques

Instruction ***DIV***

- But: diviser le contenu du couple de registre **DX-AX** par le contenu de l'opérande source et placer le quotient dans **AX** et le reste dans **DX**
- Syntaxe:

  DIV source

  (rem : *source* ne peut pas être une valeur immédiate!)
- Flags:

  tous modifiés mais indéfinis
- Exemple:

  DIV CX (AX <- DX-AX / CX et DX <- reste de la division)

Les instructions arithmétiques

Instruction ***INC***

- But : L'instruction **INC** incrémente d'une unité l'opérande de destination
- Syntaxe:

  INC destination

(rem : *destination* ne peut pas être une valeur immédiate!).

- Flags:

  - **S** est mis à 0 si le bit de poids fort = 0.
  - **Z** est mis à 1 si le résultat = 0.
  - **O** est mis à 0 si le résultat peut être installé dans l'opérande destination (pas de retenue sur les 2 derniers bits).

Exemple : INC AX (AX← AX+1)

Les instructions comparaison

Instruction ***CMP*** :

- But : comparer (soustraire) le contenu de l'opérande source au contenu de l'opérande destination.
- Syntaxe :

  CMP *destination, source*

- Flags :

  - **O** est mis à 0
  - **S** est mis à 0 si le bit de signe (à gauche) = 0
  - **Z** est mis à 0 si le résultat est différent de 0.
  - **C** est mis à 1 s"il y a retenue.

Instructions de Branchement

Le microprocesseur utilise deux registres pour connaître la prochaine instruction à exécuter : CS (code segment) et IP (déplacement du premier byte de l'instruction). L'adresse complète est donc CS:IP.

- Il existe 5 types d'instructions de rupture de séquence:
  - les sauts ***inconditionnels*** (JMP);
  - les sauts ***conditionnels*** (JZ, JL, ...);
  - les boucles (LOOP);
  - les sous-programmes (CALL);
  - les interruptions (INT).

Instructions de Branchement

En réalité : nécessité de programme **non** séquentiel:

- Faire 10 fois
- Si ... Alors ... Sinon ...
- Répéter jusqu'à...
- Tant que ... faire ...

Sur processeur: (et donc en assembleur)

- On peut "aller à" une instruction "marquée" (via un label).
- Ce branchement à une instruction peut se faire conditionnellement ou inconditionnellement.
- **Les conditions se font sur l'état des flags**
- Une seule exception : test sur CX = 0 (« pour Loop")

Instructions de Branchement

Instruction ***JMP***

- But : "se brancher" **inconditionnellement** à l'instruction marquée par label:
- Syntaxe :

  JMP label

  ...

  label: **...**
- Flags :

  aucun flag n'est modifié

Instructions de Branchement

Instruction ***LOOP***

- But : "se brancher" à l'instruction marquée par label: tant que CX#0
- Syntaxe :

  MOV *CX,17*
  *label:**...***
  ...
  LOOP *label*

- Flags :

  aucun flag n'est modifié.

- Remarque :
  - Loop permet de répéter plusieurs fois la même séquence d'instructions.
  - A chaque passage dans la boucle, le registre CX est ***décrémenté*** d'une unité et, tant qu'il n'est pas égal à 0, le programme saute à l'adresse spécifiée par le label.

Instructions de Branchement

Branchements conditionnels avec *cmp*:	
Instruction	**État du flag**
JN**C**	Branchement si C=0
J**C**	C=1
JN**O**	O=0
J**O**	O=1
JN**S**	S=0
J**S**	S=1
JN**Z**	Z=0
J**Z**	Z=1

Instructions de Branchement

Tables des branchements conditionnels			
Non Signés			
	Instr.	*Signification*	*Flags*
>	JA / JNBE	(above) / (not below or equal)	c=0 et z=0
>=	JAE / JNB	(above or equal) / (not below)	c=0
<	JB / JNAE	(below) / (not above or equal)	c=1
<=	JBE / JNA	(below or equal) / (not above)	c=1 et z=1

Instructions de Branchement

SIGNES			
	Instruction	*Signification*	*flags*
>	JG /JNLE	(greater) / (not less or equal	z=0 et s=**o**
>=	JGE /JLE	(greater or equal) / (not less)	s=**o**
<	JL /JNGE	(less) / (not greater or equal)	s# **o**
<=	JLE / JNG	(less or equal) / (not greater)	c=1 et s#**o**
POUR LES DEUX			
=	JE	equal	z=1
#	JNE	not equal	z=0

À faire :

- *Écrire les programmes assembleur qui traduisent les expressions suivantes:*
  1. *Si (AX>=BX) Alors CX=1 Sinon CX=0;*
  2. *Si (AX< BX) Alors CX=AX Sinon CX=BX;*
  3. *Si (AX=BX) Alors CX=0 Sinon CX=1;*
  4. *Si (AX>=BX) Alors*

     CX ← 10

     tant que CX # 0 faire AX ← AX+1

     fin tantque

     sinon CX ← 0;