TẬP LỆNH X86

- **ADD,**
- **SUB,CMP, AND, TEST, OR, XOR**

  REG, memory
  memory, REG
  REG, REG
  memory, immediate
  REG, immediate

  REG: AX, BX, CX, DX, AH, AL, BL, BH, CH, CL, DH, DL, DI, SI, BP, SP.

  memory: [BX], [BX+SI+7], variable, ...

  immediate: 5, -24, 3Fh, 10001101b, ..

- **CF**, **ZF**, **SF**, **OF**, **PF**, **AF**.

TẬP LỆNH X86

- **ADD**
- **ADC dst, src**

dst = dst + src + CF

adc reg, reg
adc reg, mem
adc reg, imm
adc mem, reg
adc mem, imm

HỢP NGỮ

- **SUB dst, src**

  dst = dst – src

- **CMP dst, src (compare)**

  so sánh dst và src

  dst – src, không thay đổi dst, src

HỢP NGỮ

- **SBB dst, src**

  dst = dst – src – CF

HỢP NGỮ

- **CMP dst, src (compare)**

  so sánh dst và src

  dst – src, không thay đổi dst, src

  cmp reg, reg

  cmp reg, mem

  cmp reg, imm

  cmp mem, reg

  cmp mem, imm

HỢP NGỮ

- **OR dst, src**

  dst = dst | src

  phép OR bit hệ nhị phân

  or reg, reg

  or reg, mem

  or reg, imm

  or mem, reg

  or mem, imm

OR bit : kiểm tra số khác không hay bằng 0.

Nếu bằng 0, OR lai chính nó, kết quả =0 =>ZF =1

- Set bit,

HỢP NGỮ

- **XOR dst, src**

  dst = dst ^ src

  phép XOR bit hệ nhị phân

  xor reg, reg

  xor reg, mem

  xor reg, imm

  xor mem, reg

  xor mem, imm

XOR bit : xóa về 0.

XOR bit: toggle bit

HỢP NGỮ

AND dst, src

dst = dst & src

phép AND bit hệ nhị phân

and reg, reg

and reg, mem

and reg, imm

and mem, reg

and mem, imm

⇒ **AND bit : kiểm tra số chẵn hay số lẻ, số âm , số ko âm.**

⇒ **Xóa bit về 0.**

HỢP NGỮ

TEST dst, src

dst & src

thực hiện phép AND bit hệ nhị phân, không thay đổi dst, src

test reg, reg

test reg, mem

test reg, imm

test mem, reg

test mem, imm

HỢP NGỮ

- **INC dst**

tăng 1

dst = dst +1

- **DEC dst**

dst = dst -1

HỢP NGỮ

- **NEG dst**

Bù 2 của dst, đảo dấu

dst = -dst

HỢP NGỮ

- **NOT dst**

Bù 1 của dst, đảo bit của dst

dst = đảo bit = 2^n -1 –dst

HỢP NGỮ

- **SHL dst, 1**
- **SHL dst, CL**

dịch trái CL bit, (a<<k = a * 2 ^k)

SHL reg, CL
SHL mem, CL
SHL reg, 1
SHL mem, 1

HỢP NGỮ

- **SHR dst, 1**
- **SHR dst, CL**

dịch phải CL bit, (a>>k = a / 2 ^k)

SHR reg, CL
SHR mem, CL
SHR reg, 1
SHR mem, 1

HỢP NGỮ

- **MUL**
- **DIV**

- **IMUL : tuong tự MUL , nhân có dấu**

- **IDIV: tương tự DIV, chia có dấu.**

HỢP NGỮ

- **ADD, SUB,CMP, AND, TEST, OR, XOR**

  REG, memory
  memory, REG
  REG, REG
  memory, immediate
  REG, immediate

  REG: AX, BX, CX, DX, AH, AL, BL, BH, CH, CL, DH, DL, DI, SI, BP, SP.

  memory: [BX], [BX+SI+7], variable, ...

  immediate: 5, -24, 3Fh, 10001101b, ..

- **CF**, **ZF**, **SF**, **OF**, **PF**, **AF**.

HỢP NGỮ

- **MUL, IMUL, DIV, IDIV**

  REG
  memory

  REG: AX, BX, CX, DX, AH, AL, BL, BH, CH, CL, DH, DL, DI, SI, BP, SP.

  memory: [BX], [BX+SI+7], etc...

- **CF, OF**

HỢP NGỮ

- **MUL - Unsigned multiply**
  - **when operand is a byte:**
    AX = AL * operand.
  - **when operand is a word:**
    (DX AX) = AX * operand.

HỢP NGỮ

- **IMUL - Signed multiply:**
  - **when operand is a byte:**
    AX = AL * operand
  - **when operand is a word:**
    (DX AX) = AX * operand

HỢP NGỮ

- **DIV - Unsigned divide:**
  - **when operand is a byte:**
    AL = AX / operand
    AH = remainder (modulus).
  - **when operand is a word:**
    AX = (DX AX) / operand
    DX = remainder (modulus).

HỢP NGỮ

- **IDIV - Signed divide:**
  - **when operand is a byte:**
    AL = AX / operand
    AH = remainder (modulus)
  - **when operand is a word:**
    AX = (DX AX) / operand
    DX = remainder (modulus)

HỢP NGỮ

- **INC, DEC, NOT, NEG**

  REG
  memory

  REG: AX, BX, CX, DX, AH, AL, BL, BH, CH, CL, DH, DL, DI, SI, BP, SP.

  memory: [BX], [BX+SI+7], variable, etc...

  INC, **DEC** : **ZF**, **SF**, **OF**, **PF**, **AF**.

  NEG : **CF**, **ZF**, **SF**, **OF**, **PF**, **AF**.

  NOT instruction does not affect any flags!

HỢP NGỮ

- **INC, DEC, NOT, NEG**

  REG
  memory

  REG: AX, BX, CX, DX, AH, AL, BL, BH, CH, CL, DH, DL, DI, SI, BP, SP.

  memory: [BX], [BX+SI+7], variable, etc...

  INC, **DEC** : **ZF**, **SF**, **OF**, **PF**, **AF**.

  NEG : **CF**, **ZF**, **SF**, **OF**, **PF**, **AF**.

  NOT instruction does not affect any flags!

HỢP NGỮ

- **unconditional jumps**
  - JMP label

```
  jmp calc ; go to 'calc'.
back:
  jmp stop ; go to 'stop'.
calc:
  add ax, bx ; add bx to ax.
  jmp back ; go 'back'.
stop:
```

HỢP NGỮ

- **Lệnh nhảy không điều kiện**
- **TÊNNHÃN:**

  - JMP TÊNNHÃN

HỢP NGỮ

- **Lệnh nhảy không dấu**

  j : jmp
  e: equal
  n : not
  a : above
  b : below

HỢP NGỮ

Je/jz : nhảy nếu bằng
Jne/Jnz: nhảy nếu khác
Ja/jnbe: nhảy nếu lớn hơn
Jae/jnb: nhảy nếu lớn hơn hoặc bằng
Jb/jnae: nhảy nếu nhỏ hơn
Jbe/jna: nhảy nếu nhỏ hơn hoặc bằng

HỢP NGỮ

- **Lệnh nhảy có dấu**

  j : jmp
  e: equal

  n : not

  g : greater

  l : less

HỢP NGỮ

Je/jz : nhảy nếu bằng
Jne/Jnz: nhảy nếu khác
Jg/jnle: nhảy nếu lớn hơn
Jge/jnl: nhảy nếu lớn hơn hoặc bằng
Jl/jnge: nhảy nếu nhỏ hơn
Jle/jng: nhảy nếu nhỏ hơn hoặc bằng

Biến đổi các phát biểu cấu trúc sang hợp ngữ

- If ; if …. else
- If and else
- If or else
- Switch …case => if
- While
- For
- Do … while