1. tập Lệnh trong Assembly
- add $1, $2, $3
cộng một hằng số: addi $1,$2,4
- sub $1,$2,$3
2. Load dữ liệu từ bộ nhớ lên thanh ghi
lw $r1, 100 ,$r2 ⟸⟹ r1 = mem[100 + r2]($r2: base register; 100: offset)
3.để chuyện dữ liệu từ 1 thanh ghi suống bộ nhớ
sw $r1,100,$r2 <==> mem[100+r2]=r1
4. lệnh so sánh
beq $r1, $r2, label
nếu r1=r2 thì nhảy đến label, ngược lại thì thực hiện lênh tiếp theo
bne $r1, $r2, label
nếu r1!=r2 thì nhảy đến label, ngược lại thì thực hiện lênh tiếp theo
5. j label ⟸⟹ jump to label
jr $ra ⟸⟹ jump to ra
6. Thao tác logic
- and $1,$2,$3
- or $1,$2,$3
- andi $1,$2,const
- ori $1,$2,const
- sll $1,$2, const
- srl $1,$2, const
7. lệnh mov
move $8, $18 ⟸⟹ add $8, $zero, $18
8.
.text:

-viết các lệnh sau chỉ thị này
-Đoạn này chứ mã assembly thực. Được bắt đầu với khai báo global_start, khai báo này nói với kernel là chương trình sẽ thực thi ở đâu.

.data : -khai báo biến sau chỉ thị này

-Chứa data không thay đổi tại lúc runtime
-Khai báo data và được khởi tạo giá trị.

.globl L: nói cho assember là nhãn L toàn cục, có thể tham khảo từ một file khác hơn là nó tự định nghĩa

.exterm L: L mô tả một nhãn toàn cục của một file khác.
9. lệnh cmp
- cmp vleft , vright

10.phép so sánh, cấu trúc điều khiển
- cmp vleft, vright
1) so sánh 2 số nguyên dương
- ta sử dụng cơ ZF(zero flag), CF(carry flag)
- ZF = 1 khi vleft= vright
- CF: if(vleft>vright) {CF = 0; ZF=0;}
ìf(vleft<vright) {CF=1; ZF=0}

2, so sánh 2 số nguyên có dấu

- ta sử dụng 3 cờ là : ZF,OF(overflows flag : cờ tràn),SF(sign flag: cờ dấu)
- if(vleft=vright) ZF=1;
- if(vleft>vright) {
    ZF=0;
    SF=OF;
  }
-if(vleft<vright){
    ZF=0;
    SF!=OF;
  }

Instruction	Content
JZ	branches only if ZF is set
JNZ	branches only if ZF is unset
JO	branches only if OF is set
JNO	branches only if OF is unset
JS	branches only if SF is set
JNS	branches only if SF is unset
JC	branches only if CF is set
JNC	branches only if CF is unset
JP	branches only if PF is set
JNP	branches only if PF is unset

Example:

1).

```
if ( EAX == 0 )
    EBX = 1;
else
EBX = 2;
```

could be written in assembly as:

```
cmp  eax, 0                ; set flags (ZF set if eax - 0 = 0)
jz   thenblock             ; if ZF is set branch to thenblock
mov  ebx, 2                ;ELSE part of IF
jmp  next ;                ; THEN part of IF
thenblock:
mov  ebx, 1          ;jump over THEN part of IF
next:
```

2)

```
if ( EAX >= 5 )
    EBX = 1;
else
```

EBX = 2;

could be written in assembly as:

```
        cmp     eax, 5          ; goto signon if SF = 1
        js      signon          ; goto elseblock if OF = 1 and SF = 0
        jo      elseblock       ;goto thenblock if SF = 0 and OF = 0
        jmp     thenblock       ; goto thenblock if SF = 1 and OF = 1
signon:
        jo      thenblock       ; goto thenblock if SF = 1 and OF = 1
elseblock:
        mov     ebx, 2
        jmp     next
thenblock:
        mov     ebx, 1
next:
```

Signed	Signed	Unsigned	Unsigned
JE	branches if vleft = vright	JE	branches if vleft = vright
JNE	branches if vleft != vright	JNE	branches if vleft != vright
JL, JNGE	branches if vleft < vright	JB,JNAE	branches if vleft < vright
JLE, JNG	branches if vleft <= vright	JBE,JNA	branches if vleft <= vright
JG, JNLE	branches if vleft > vright	JA, JNBA	branches if vleft > vright
JGE, JNL	branches if vleft >= vright	JAE, JNB	branches if vleft >= vright

```
        cmp             eax, 5
        jge             thenblock
        mov             ebx, 2
        jmp             next
thenblock:
        mov     ebx, 1
next:
```

branch and loop
indirect jump

11. Loop instructions
11.1

- LOOP Decrements ECX, if ECX = 0, branches to label
- LOOPE, LOOPZ Decrements ECX (FLAGS register is not modified), if ECX = 0 and ZF = 1, branches
- LOOPNE, LOOPNZ Decrements ECX (FLAGS unchanged), if ECX = 0 and ZF = 0, branches

example:

-C language:

```
sum = 0;
for ( i =10; i >0; i −− )
        sum += i;
```

could be translated into assembly as:

```
        mov         eax, 0          ; eax is sum
        mov         ecx, 10                 ; ecx is i
loop_start:
        add         eax, ecx
        loop        loop_start
```

11.2

```
while( condition ) {
        body of loop;
}
```

```
while:
        ; code  to set FLAGS based on condition
        jxx     endwhile        ; select xx so that branches if false
        ; body of loop
        jmp     while
endwhile:
```

11.3

```
do {
        body of loop;
} while( condition );
```

this could be translated into:

```
do:
        ;body of loop
        ;code to set FLAGS based on condition
        jxx     do      ;select xx so that branches if true
```

(link tham khao)
http://www.cs.virginia.edu/~evans/cs216/guides/x86.html