Topics

- Unconditional Jumps
- Conditional jump and compare instructions
- Implementing Loop Structures
- FOR Loops
- Arrays

Conditional Jumps, Compare Instructions and if Structures

Conditional Jump Instructions

- Format: `j--  targetStatement`
- The last part of the mnemonic identifies the condition under which the jump is to be executed.
  - If the condition holds, then the jump takes place and the next statement executed is at *targetStatement:*
  - Otherwise, the next instruction (the one following the conditional jump) is executed.
- Used to implement *if* structures, other selection structures, and loop structures in 8ox86 assembly language

je---> if equal to

jne---> if not equal to

jz----> if zero

jnz---> if non zero

```
  mov EDX, 45
  mov ECX, 20
  cmp EDX, ECX
  jle JUMPLABEL2
  mov EAX, 1
  ret
JUMPLABEL2:
  mov EAX, -1
  ret
```

edx<=ecx
45 20
so false

Conditional Jumps and Flags

- Most "conditions" considered by the conditional jump instructions are settings of flags in the EFLAGS register.
- Example

```
    jz endWhile
```

  means to jump to the statement with label *endWhile* if the zero flag ZF is set to 1
- Conditional jump instructions don't modify flags; they react to previously set flag values.

based on flag registers jump instruction will decide whether condition is true or false

cmp Instructions

- Most common way to set flags for conditional jumps
- Flag
  - Set: Corresponding bit is 1 (True)
  - Reset: Corresponding bit is 0 (False)
- Format: `cmp` *`operand1, operand2`*
  - Flags are set the same as for the subtraction operation ***operand1 – operand2***
  - Operands are not changed

cmp Examples

				Flags				Interpretation	
#	Op1	Op2	Diff	CF	OF	SF	ZF	Signed	Unsigned
1	3B	3B	00	0	0	0	1	Op1=Op2	Op1=Op2
2	3B	15	26	0	0	0	0	Op1>Op2	Op1>Op2
3	15	3B	DA	1	0	1	0	Op1<Op2	Op1<Op2
4	F9	F6	03	0	0	0	0	Op1>Op2	Op1>Op2
5	F6	F9	FD	1	0	1	0	Op1<Op2	Op1<Op2
6	15	F6	1F	1	0	0	0	Op1>Op2	Op1<Op2
7	F6	15	E1	0	0	1	0	Op1<Op2	Op1>Op2
8	68	A5	C3	1	1	1	0	Op1>Op2	Op1<Op2
9	A5	68	3D	0	1	0	0	Op1<Op2	Op1>Op2

F9-->1111 1001 1-->-ve 0--->+ve MSB ----> most significant bit

cmp Encoding

Op1	Op2	Opcode	#bytes of obj code
register 8	Immediate 8	80	3
register 16	Immediate 8	83	3
register 32	Immediate 8	83	3
register 16	Immediate 16	81	4
register 32	Immediate 32	81	6
AL	Immediate 8	3C	2
AX	Immediate 16	3D	3
EAX	Immediate 32	3D	5
mem byte	Immediate 8	80	3+
mem word	Immediate 8	83	3+
mem doubleword	Immediate 8	83	3+
mem word	Immediate 16	81	4+
mem doubleword	Immediate 32	81	6+

cmp Encoding

Op1	Op2	Opcode	#bytes of obj code
register 8	register 8	3A	2
register 16	register 16	3B	2
register 32	register 32	3B	2
register 8	mem byte	3A	2+
register 16	mem word	3B	2+
register 32	mem doubleword	3B	2+
mem byte	register 8	38	2+
mem word	register 16	39	2+
mem doubleword	register 32	39	2+

cmp Encoding

Op1	Op2	Opcode	#bytes of obj code
register 8	register 8	3A	2
register 16	register 16	3B	2
register 32	register 32	3B	2
register 8	mem byte	3A	2+
register 16	mem word	3B	2+
register 32	mem doubleword	3B	2+
mem byte	register 8	38	2+
mem word	register 16	39	2+
mem doubleword	register 32	39	2+

cmp Encoding – 64 bit mode

Op1	Op2	Opcode	#bytes of obj code
register 64	Immediate 8	83	3
register 64	Immediate 32	81	6
RAX	Immediate 32	3D	5
mem quadword	Immediate 8	83	3+
mem quadword	Immediate 32	81	6+
register 64	register 64	3B	2
register 64	mem quadword	3B	2+
mem quadword	register 64	39	2+

cmp Scenarios for conditional jumps

Appropriate mnemonic to be used based on signedness and criteria.

- Conditional Jumps to Use After Signed Operand Comparison
- Conditional Jumps to Use After Unsigned Operand Comparison
- Other Conditional Jumps (signedness does not matter)

Conditional Jumps – mnemonics/code

Instruction	Description	signed-ness	Flags	Short jump opcodes	near jump opcodes
JO	Jump if overflow		OF = 1	70	0F 80
JNO	Jump if not overflow		OF = 0	71	0F 81
JS	Jump if sign		SF = 1	78	0F 88
JNS	Jump if not sign		SF = 0	79	0F 89
JE **JZ**	Jump if equal Jump if zero		ZF = 1	74	0F 84
JNE **JNZ**	Jump if not equal Jump if not zero		ZF = 0	75	0F 85
JB **JNAE** **JC**	Jump if below Jump if not above or equal Jump if carry	unsigned	CF = 1	72	0F 82
JNB **JAE** **JNC**	Jump if not below Jump if above or equal Jump if not carry	unsigned	CF = 0	73	0F 83

after execution of cmp statement few flags are set

conditional jump will check those flags and decide whether condiition is true or false

Conditional Jumps - mnemonics

Instruction	Description	signed-ness	Flags	Short jump opcodes	near jump opcodes
JBE **JNA**	Jump if below or equal Jump if not above	unsigned	CF = 1 or ZF = 1	76	0F 86
JA **JNBE**	Jump if above Jump if not below or equal	unsigned	CF = 0 and ZF = 0	77	0F 87
JL **JNGE**	Jump if less Jump if not greater or equal	signed	SF <> OF	7C	0F 8C
JGE **JNL**	Jump if greater or equal Jump if not less	signed	SF = OF or ZF =1	7D	0F 8D
JLE **JNG**	Jump if less or equal Jump if not greater	signed	ZF = 1 or SF <> OF	7E	0F 8E
JG **JNLE**	Jump if greater Jump if not less or equal	signed	ZF = 0 and SF = OF	7F	0F 8F

Conditional Jumps - mnemonics

Instruction	Description	signed-ness	Flags	Short jump opcodes	near jump opcodes
JP **JPE**	Jump if parity Jump if parity even		PF = 1	7A	0F 8A
JNP **JPO**	Jump if not parity Jump if parity odd		PF = 0	7B	0F 8B
JCXZ **JECXZ**	Jump if CX register is 0 Jump if ECX register is 0		CX = 0 ECX = 0	E3	

Example Usage

- Example-1

```
cmp   eax, nbr
jle   smaller
```

  - The jump will occur if the value in EAX is less than or equal than the value in *nbr*, where both are interpreted as signed numbers.

- Example-2

```
  mov EDX, 45
  mov ECX, 20
  cmp EDX, ECX
  jle JUMPLABEL2
  mov EAX, 1
  ret
JUMPLABEL2:
  mov EAX, -1
  ret
```

```
000000C5  BA 0000002D          mov EDX, 45
000000CA  B9 00000014          mov ECX, 20
000000CF  3B D1             cmp EDX, ECX
000000D1  7E 06             jle JUMPLABEL2
000000D3  B8 00000001          mov EAX, 1
000000D8  C3                ret
000000D9               JUMPLABEL2:
000000D9  B8 FFFFFFFF          mov EAX, -1
000000DE  C3                ret
```

45<=20----> false

edx-->45
ecx-->20

cmp edx,ecx

jle jumplabel2

if Example-1

Assumptions

- *value* in EBX
- *smallCount* and *largeCount* in memory

Design

if value < 10
then
  add 1 to smallCount;
else
  add 1 to largeCount;
end if;

Code

```
            cmp   ebx, 10
            jnl   elseLarge
            inc   smallCount
            jmp   endValueCheck
elseLarge:  inc   largeCount
endValueCheck:
```

cmp ebx,10
        jl small
        inc largecount
        jmp endvaluecheck

small: inc smallcount
endvaluecheck:

if value is not less than 10
increment large count
else
increment smallcount

if Example-2 – *or* logic

Assumptions

- *total* and *value* in memory
- *count* in ECX

Design

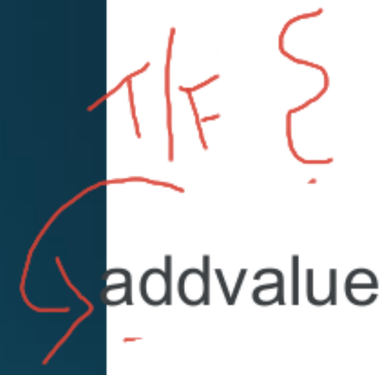
if (total ≥ 100)
or (count = 10)
then
add value to total;
end if;

Code

```
            cmp   total, 100
            jge   addValue
            cmp   ecx, 10
            jne   endAddCheck
addValue:   mov   ebx, value
            add   total, ebx
endAddCheck:
```

```
cmp total,100
        jge addvalue
        cmp ecx,10
        je  addvalue

addvalue:mov ebx,value
        add total,ebx
endaddcheck:
```



wrong

jump is true control shifts to the label provided

jump is false , control goes to the instruction which is next to jump

if Example-3 – *and* logic

Assumptions

- *total* and *value* in memory
- *count* in ECX

Design

if (total ≥ 100)
and (count = 10)
then
add value to total;
end if;

Code

```
            cmp   total, 100
            jnge  endAddCheck
            cmp   ecx, 10
            jne   endAddCheck
addValue:   mov   ebx, value
            add   total, ebx
endAddCheck:
```

if total not greater than or equal to 100 and count not equal to 10
come out of logic
else
add value to total