6. Conditional Processing

Tuesday, September 10, 2024 12:22 PM

6.2 Conditional Jumps

CMP Instruction

→ compares leftOp and rightOp

```
CMP  reg, reg
     reg, mem
     reg, imm
     mem, reg
     Mem, imm
```

} same size operands

→ Compare instruction will followed by a conditional jump

je — jump if equal

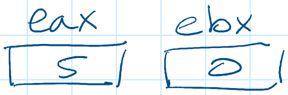
jne — jump if not equal

jg — jump if greater than (leftOp > rightOp)

jL — jump if less than (leftOp < rightOp)

example: .code

```
cmp eax, ebx
jg  L1
   .
   .
   .
L1:
   mov ecx, 5
```



* Boolean Instructions (Bitwise Operations)

* Perform bitwise comparisons.

* AND instruction, OR instruction, XOR instruction
NOT instruction

AND

destination, source
reg, reg
reg, mem
reg, imm
mem, reg
mem, imm

No mem & mem operands

8, 16, 32 bit operands
* byte, word & dword

X	Y	AND
0	0	0
1	0	0
0	1	0
1	1	1

example:

```
mov  ah, 'r'          ; 'r' - 72h
and  ah, 1101.1111b
```

$\underbrace{1101.1111b}_{0dFh}$

'r' - 72h

	128	64	32	16	8	4	2	1
	0	1	1	1	0	0	1	0
AND	1	1	0	1	1	1	1	1
	0	1	0	1	0	0	1	0

$\boxed{0101.0010}$ → 52h - 'R'

changed (cleared bit 6)

subtracts 32 or 20h from 'r'

OR Instruction

X	Y	OR
0	0	0
1	0	1
0	1	1
1	1	1

example:

```
mov  al, 'R'          ; 'R' - 52h
or   al, 0010.0000b
```

82d or

'R' - 52h

'R' - 52h

0101.0010
OR 0010.0000

0111.0010

added 32d or 20h to 52h.

al = 72h - 'r'

XOR Instruction (Exclusive OR)

X	Y	XOR
0	0	0
1	0	1
0	1	1
1	1	0

* Can be used to encrypt a String

example

```
mov ah, 'J'   ; 4Ah
mov al, '+'   ; 2Bh
XOR ah, al
XOR ah, al → '+'
```

encrypts the character → XOR ah, al

'J' - 4Ah original message

0100.1010
XOR 0010.1011 ~ '+' - 2Bh key

0110.0001
8 4 2 1 8 4 2 1

61h - 'a' — encrypted message

encryption

0110.0001 - 'a' = 61h
XOR 0010.1011 '+' - 2Bh

0100.1010 - 'J' - 4Ah

Decryption

Homework

Encrypt.

message = "big bang", 0
Key = {12h, 34h, 56h, 78h, 9Ah, ABh, BCh}

ment

Encrypt.

	'b'	'i'	'g'		'b'	'a'	'n'	'g'
XOR	12h	34h	56h	78h	9Ah	ABh	BCh	12h ← first element in keys array

NOT Instruction

```
mov  al, 80
not  al
inc  al
```

} performed two's complement on 80

128 64 32 16 8 4 2 1

NOT 0101 0000 — 80

inc (add 1) 1010 1111

+ 1

1011.0000 → -80 signed integers

80 + (-80) = 0 → 1 0101.0000

1011.0000

$\boxed{0000.0000}$