Chapter 6:
Conditional Processing

Chapter Overview

- Boolean and Comparison Instructions
- Conditional Jumps
- Conditional Loop Instructions
- Conditional Structures

6.2 Boolean and Comparison Instructions

- Boolean instructions

Instructions	동작
AND dst, src	dst ← dst AND src
OR dst, src	dst ← dst OR src
XOR dst, src	dst ← dst XOR src
NOT dst	dst ← NOT dst

src: source operand
dst: destination operand

  - bitwise Boolean operations
  - operand rule: MOV, 산술연산과 같음

Status Flags - Review

- CPU Status Flags

Flags	동작
ZF (zero)	결과가 0이면 set
SF (sign)	음수이면 set (MSB와 같음)
CF (carry)	unsigned 결과가 표현범위 벗어나면 set
OF (overflow)	signed 결과가 표현범위 벗어나면 set
PF (parity)	결과의 1값을 갖는 bit수가 짝수이면 set
AF (auxiliary)	하위4비트에 대한 carry발생하면 set

AND and OR Instruction

- AND
  - selective clear (mask operation)

```
    00111011
AND 00001111
    --------
    00001011
```

cleared — unchanged

AND

x	y	x ∧ y
0	0	0
0	1	0
1	0	0
1	1	1

- OR
  - selective set

```
    00111011
OR  00001111
    --------
    00111111
```

unchanged — set

OR

x	y	x ∨ y
0	0	0
0	1	1
1	0	1
1	1	1

XOR and NOT Instruction

- XOR
  - selective complement(invert)

```
    00111011
XOR 00001111
    --------
    00110100
```

unchanged — inverted

XOR

x	y	x ⊕ y
0	0	0
0	1	1
1	0	1
1	1	0

- NOT
  - 1's complement

```
NOT 00111011
    --------
    11000100
```

inverted

NOT

X	¬X
F	T
T	F

Example

- 대소문자 변환
  - ASCII code:
    - 01000001 ~ 01011010 A ~ Z
    - 01100001 ~ 01111010 a ~ z
  - 소문자 → 대문자 변환: bit 5를 0으로 바꿈
    대문자 → 소문자 변환: bit 5를 1로 바꿈

```
mov al,'a'                  ; AL = 01100001b
and al,11011111b            ; AL = 01000001b
```

```
mov al,'A'                  ; AL = 01000001b
or  al,00100000b            ; AL = 01100001b
```


Example

- 키보드 CapsLock 제어 (real mode용)
  - 0040:0017h (417h번지)가 Keyboard Flag 상태를 나타내는 byte임
  - bit 6에 의해서 CapsLock 상태가 정해짐
    - 0: OFF, 1: ON
  - 이 제어는 real mode에서만 동작할 수 있으며 virtual 8086 mode에서 동작하지 않음 (Windows XP, NT, 2000 에서 사용 불가능)

```
mov ax,40h                      ; BIOS segment
mov ds,ax
mov bx,17h                      ; keyboard flag byte
or BYTE PTR [bx],01000000b      ; CapsLock on
```

TEST와 CMP Instruction

- TEST dst, src
  - dst ∧ src 연산 (AND연산) 수행 (결과를 저장하지 않음)
- CMP dst, src
  - dst – src 연산 수행 (결과를 저장하지 않음)
- TEST와 CMP instruction의 공통점
  - 연산 결과를 저장하지 않고 flag에만 영향을 줌 – dst는 변하지 않음
  - 주로 conditional jump와 함께 사용

6.3 Conditional Jumps

- *J*cond Instruction
  - 형식: J*cond* label
  - 이전의 연산 결과가 주어진 조건을 만족하면 label 위치의 instruction으로 jump (이전의 연산결과는 주로 flag상태를 참조함)
- Condition 종류와 conditional jumps

Condition 종류	Instructions
특정 Flag상태	JZ, JNZ, JC, JNC, JO, JNO, JS, JNS, JP, JNP
equality	JE, JNE, JCXZ, JECXZ
signed 비교	JG, JGE, JL, JLE (Greater, Less) JNLE, JNL, JNGE, JNG
unsigned 비교	JA, JAE, JB, JBE (Above, Below) JNBE, JNB, JNAE, JNA

  - JZ와 JE, JNZ와 JNE는 같은 명령어

Jumps based on Specific Flags

Mnemonic	Description	Flags
JZ	Jump if zero	ZF = 1
JNZ	Jump if not zero	ZF = 0
JC	Jump if carry	CF = 1
JNC	Jump if not carry	CF = 0
JO	Jump if overflow	OF = 1
JNO	Jump if not overflow	OF = 0
JS	Jump if signed	SF = 1
JNS	Jump if not signed	SF = 0
JP	Jump if parity (even)	PF = 1
JNP	Jump if not parity (odd)	PF = 0

Jumps based on Equality

Mnemonic	Description
JE	Jump if equal (*leftOp* = *rightOp*)
JNE	Jump if not equal (*leftOp* ≠ *rightOp*)
JCXZ	Jump if CX = 0
JECXZ	Jump if ECX = 0

Jumps based on Unsigned Comparisons

Mnemonic	Description
JA	Jump if above (if *leftOp* > *rightOp*)
JNBE	Jump if not below or equal (same as JA)
JAE	Jump if above or equal (if *leftOp* >= *rightOp*)
JNB	Jump if not below (same as JAE)
JB	Jump if below (if *leftOp* < *rightOp*)
JNAE	Jump if not above or equal (same as JB)
JBE	Jump if below or equal (if *leftOp* <= *rightOp*)
JNA	Jump if not above (same as JBE)

Jumps Based on Signed Comparisons

Mnemonic	Description
JG	Jump if greater (if *leftOp* > *rightOp*)
JNLE	Jump if not less than or equal (same as JG)
JGE	Jump if greater than or equal (if *leftOp* >= *rightOp*)
JNL	Jump if not less (same as JGE)
JL	Jump if less (if *leftOp* < *rightOp*)
JNGE	Jump if not greater than or equal (same as JL)
JLE	Jump if less than or equal (if *leftOp* <= *rightOp*)
JNG	Jump if not greater (same as JLE)

Application – 크기 비교

- 동등 비교

```
cmp eax,ebx      ; eax - ebx
je  L1
```

if (eax==ebx)
  goto L1

- 크기 비교
  - unsigned 비교

```
cmp eax,ebx
ja  Larger
```

  - signed 비교

```
cmp eax,ebx
jg  Larger
```

if (eax>ebx)
  goto Larger

Application – 비트 검사

- 비트검사
  - AL의 bit 0 또는 bit 1이 1이면 jump

```
test al,00000011b
jnz L1
```

  - AL의 bit 0과 bit 1이 모두 0이면 jump

```
test al,00000011b
jz L1
```

  - AL의 bit 0과 bit 1이 모두 1이면 jump (AL 내용 변경)

```
and al,00000011b        ; clear unwanted bits
cmp al,00000011b        ; check remaining bits
je  L1                  ; all set? jump to L1
```

Application – 짝수, Zero 검사

- 짝수 검사
  - if (wordVal is even) goto L1

```
mov ax,wordVal
test ax,1           ; bit 0 set?
jz  L1              ; jump if ZF set
```

- Zero/Nonzero 검사
  - if (AL ≠ 0) goto L1

```
or  al,al      ; AL unchanged, Flag 영향
jnz L1         ; jump if not zero
```

  - if (AL = 0) goto L1

```
or  al,al      ; AL unchanged
jz L1          ; jump if zero
```


Application – 최대값 찾기

- v1, v2, v3에 있는 unsigned 값의 최대값을 max에 저장

```
.data
v1   word 10
v2   word 50
v3   word 30
max  word ?
.code
     mov ax,v1        ; ax=v1 (assume v1 is larger)
     cmp ax,v2
     jae L1
     mov ax,v2        ; if (ax<v2)ax = v2
L1:  cmp ax,v3
     jae L2
     mov ax,v3        ; if (ax<v3)ax = v3
L2:  mov max,ax       ; max = ax
```


배열 스캔 – 조건을 만족하는 원소 찾기

- 0이 아닌 원소를 찾으면 배열 스캔 종료

```
.data
array SWORD 0,0,0,1,20,0,-1
.code
    mov ebx, OFFSET array          ; array address
    mov ecx, LENGTHOF array        ; loop counter
L1:cmp WORD PTR [ebx], 0
    jnz found
    add ebx, 2
    loop L1
    jmp notfound
found: ...
    jmp quit
notfound: ...
quit:
```


Bit Test Instructions

- BT dst, n
  - 동작: CF ← dst의 bit n (dst 변화 없음)
  - operand: dst는 r/m16, r/m32
    n은 imm8 또는 dst와 같은 크기의 register
- Example
  - AX의 bit 9가 1이면 jump

```
bt AX,9                ; CF = bit 9
jc L1                  ; jump if CF set
```

Bit Test Instructions

- BTC dst, n (bit test and complement)
  - 동작: CF ← dst의 bit n
    dst의 bit n ← NOT dst의 bit n (보수화)
- BTR dst, n (bit test and reset)
  - 동작: CF ← dst의 bit n
    dst의 bit n ← 0 (reset)
- BTS dst, n (bit test and set)
  - 동작: CF ← dst의 bit n
    dst의 bit n ← 1 (set)

```
.data
semaphore WORD 10001000b
.code
   bts semaphore, 7 ; CF←1, semaphore←00001000b
   jnc next
```


6.4 Conditional Loop Instructions

- LOOPZ (LOOPE) Instruction
  - 동작: ECX ← ECX – 1
    if ECX > 0 and ZF=1, jump to destination
  - 용도: 주어진 값과 일치하지 않은 첫 배열 원소를 찾음
    (ECX가 0이거나 비교결과가 같지 않을 때 루프 종료)
- LOOPNZ (LOOPNE) Instruction
  - 동작: ECX ← ECX – 1
    if ECX > 0 and ZF=0, jump to destination
  - 용도: 주어진 값과 일치하는 첫 배열 원소를 찾음
    (ECX가 0이거나 비교결과가 같을 때 루프 종료)

Example

- 양수를 찾을 때까지 배열 스캔

```
.data
array SWORD -3,-6,-1,-10,10,30,40,4
sentinel SWORD 0
.code
   mov esi,OFFSET array
   mov ecx,LENGTHOF array
next:
   test WORD PTR [esi],8000h  ; test sign bit
   pushfd                     ; push flags on stack
   add esi,2
   popfd                      ; pop flags from stack
   loopnz next                ; continue loop
   jnz quit                   ; none found
   sub esi,2                  ; ESI points to value
quit:
```


6.5 Conditional Structures

- Block-Structured IF Statements
- Compound Expressions with AND
- Compound Expressions with OR
- WHILE Loops
- Table-Driven Selection

Block-Structured IF Statements

- C언어

```
if( op1 == op2 ){
  X = 1;
  Y = 2;
} else {
  X = 2;
  Y = 1;
}
```

반대

- 어셈블리언어

```
    mov eax,op1
    cmp eax,op2
    jne L1
    mov X,1      } then block
    mov Y,2
    jmp L2
L1: mov X,2      } else block
    mov Y,1
L2:
```


Block-Structured IF Statements - 다른방법

- C언어

```
if( op1 == op2 ){
  X = 1;
  Y = 2;
} else {
  X = 2;
  Y = 1;
}
```

- 어셈블리언어

```
    mov eax,op1
    cmp eax,op2
    je L1
    mov X,2      } else block
    mov Y,1
    jmp L2
L1: mov X,1      } then block
    mov Y,2
L2:
```


Compound Expression with AND

- AND의 Short-circuit 평가
  - if (비교식1 && 비교식2 && 비교식3) …
    앞의 비교식이 거짓이면 뒤의 비교식을 실행하지 않음

```
if (AL>BL && BL>CL)
  X = 1;
```

```
    cmp al,bl    ; 비교식1
    ja  L1
    jmp next
L1:
    cmp bl,cl    ; 비교식2
    ja  L2
    jmp next
L2:
    mov X,1      ; then block
next:
```


Compound Expression with AND - 다른방법

- C언어

```
if (AL>BL && BL>CL)
  X = 1;
```

반대

- 어셈블리언어

```
    cmp al,bl    ; 비교식1
    jbe next
L1:
    cmp bl,cl    ; 비교식2
    jbe next
L2:
    mov X,1      ; then block
next:
```

Compound Expression with OR (1 of 2)

- OR의 short-circuit 평가
  - if (비교식1 || 비교식2 || 비교식3) …
    앞의 비교식이 참이면 뒤의 비교식을 실행하지 않음

```
if (AL>BL || BL>CL)
  X = 1;
```

```
    cmp al,bl    ; 비교식1
    ja  L1
    cmp bl,cl    ; 비교식2
    jbe next
L1:
    mov X,1
next:
```


WHILE Loops

- C언어 / 어셈블리언어

```
while( a < b) {
   a++;
   b--;
```

반대

```
    mov eax,a
while:
    cmp eax,b
    jnl endwhile
    inc eax
    dec b
    jmp while
endwhile:
    mov a,eax
```