Conditional Processing

COE 205

Computer Organization and Assembly Language

Dr. Aiman El-Maleh

College of Computer Sciences and Engineering

King Fahd University of Petroleum and Minerals

[Adapted from slides of Dr. Kip Irvine: Assembly Language for Intel-Based Computers]

BT Instruction

- ❖ BT = Bit Test Instruction
- ❖ Syntax:

  BT *r/m16, r16*

  BT *r/m32, r32*

  BT *r/m16, imm8*

  BT *r/m32, imm8*
- ❖ Copies bit *n* from an operand into the Carry flag
- ❖ Example: jump to label L1 if bit 9 is set in AX register

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

Next . . .

- ❖ Boolean and Comparison Instructions
- ❖ Conditional Jumps
- ❖ Conditional Loop Instructions
- ❖ Translating Conditional Structures
- ❖ Indirect Jump and Table-Driven Selection
- ❖ Application: Sorting an Integer Array

LOOPE and LOOPZ

- Syntax:

  LOOPE destination

  LOOPZ destination

- Logic:
  - ECX ← ECX – 1
  - if ECX > 0 and ZF=1, jump to destination
- Useful when scanning an array for the first element that does not match a given value.

LOOPNE and LOOPNZ

- Syntax:

  LOOPNE destination

  LOOPNZ destination

- Logic:
  - ECX ← ECX – 1;
  - if ECX > 0 and ZF=0, jump to destination
- Useful when scanning an array for the first element that matches a given value.

LOOPZ Example

The following code finds the first negative value in an array

```
.data
array SWORD 17,10,30,40,4,-5,8
.code
   mov esi, OFFSET array – 2  ; start before first
   mov ecx, LENGTHOF array    ; loop counter
L1:
   add esi, 2                 ; point to next element
   test WORD PTR [esi], 8000h ; test sign bit
   loopz L1                   ; ZF = 1 if value >= 0
   jnz found                  ; found negative value
notfound:
   . . .           ; ESI points to last array element
found:
   . . .           ; ESI points to first negative value
```

Your Turn . . .

Locate the first zero value in an array

If none is found, let ESI be initialized to -1

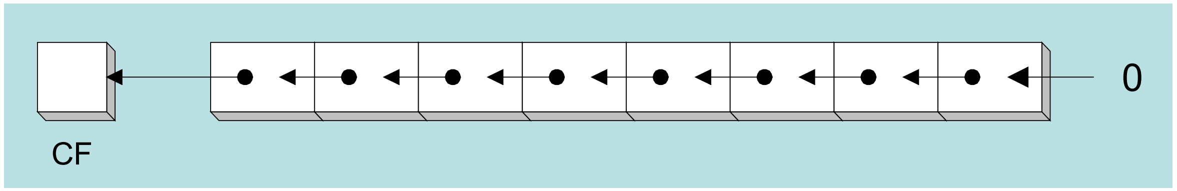
```
.data
array SWORD -3,7,20,-50,10,0,40,4
.code
   mov esi, OFFSET array – 2  ; start before first
   mov ecx, LENGTHOF array    ; loop counter
L1:
   add esi, 2                 ; point to next element
   cmp WORD PTR [esi], 0      ; check for zero
   loopne L1                  ; continue if not zero
   JE Found
   MOV ESI, -1
   Found:
```

Outline

- Shift and Rotate Instructions
- Shift and Rotate Applications
- Multiplication and Division Instructions
- Translating Arithmetic Expressions
- Decimal String to Number Conversions

SHL Instruction

- SHL is the Shift Left instruction
  - Performs a logical left shift on the destination operand
  - Fills the lowest bit with zero
  - The last bit shifted out from the left becomes the Carry Flag



- Operand types for SHL:

```
SHL reg,imm8
SHL mem,imm8
SHL reg,CL
SHL mem,CL
```

The shift count is either:

8-bit immediate ***imm8***, or

stored in register ***CL***

Only least sig. 5 bits used

Fast Multiplication

Shifting left 1 bit multiplies a number by 2

```
mov dl,5
shl dl,1
```

Before:	0 0 0 0 0 1 0 1	= 5
After:	0 0 0 0 1 0 1 0	= 10

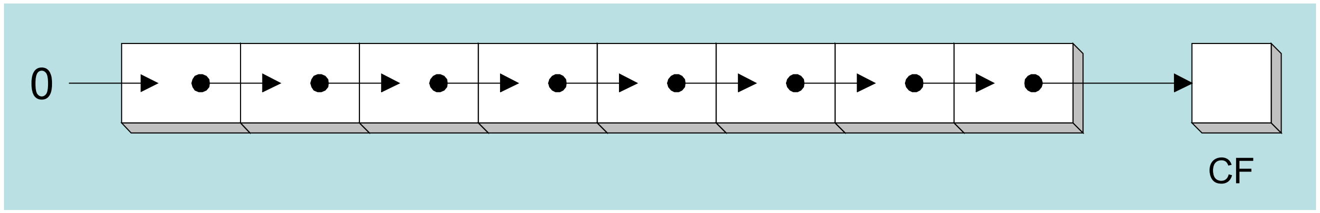
Shifting left *n* bits multiplies the operand by 2^n

For example, $5 * 2^2 = 20$

```
mov dl,5   ; DL = 00000101b
shl dl,2   ; DL = 00010100b = 20, CF = 0
```

SHR Instruction

- SHR is the Shift Right instruction
  - Performs a logical right shift on the destination operand
  - The highest bit position is filled with a zero
  - The last bit shifted out from the right becomes the Carry Flag
  - SHR uses the same instruction format as SHL



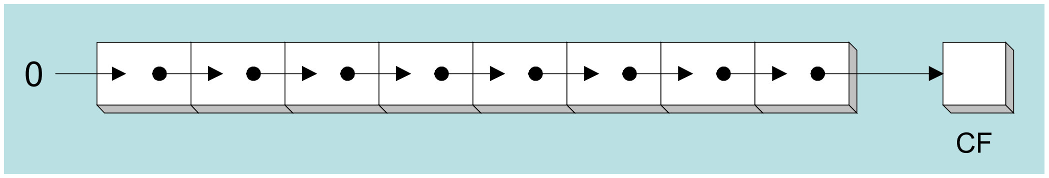
- Shifting right *n* bits divides the operand by 2^n

```
mov dl,80   ; DL = 01010000b
shr dl,1    ; DL = 00101000b = 40, CF = 0
shr dl,2    ; DL = 00001010b = 10, CF = 0
```

Logical versus Arithmetic Shifts

❖ Logical Shift

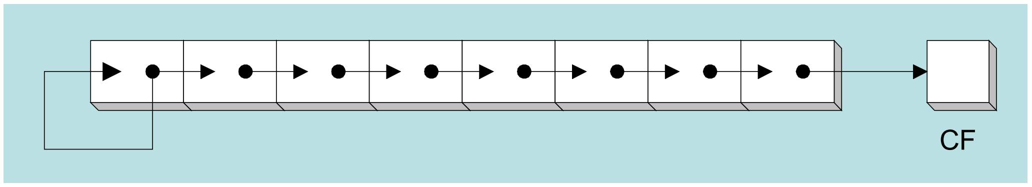
✧ Fills the newly created bit position with zero



❖ Arithmetic Shift

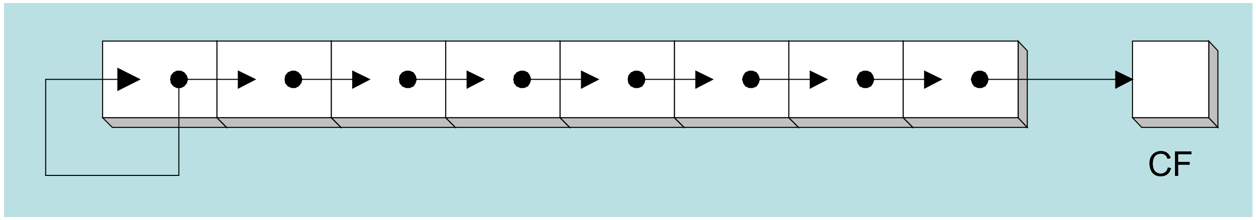
✧ Fills the newly created bit position with a copy of the sign bit

✧ Applies only to Shift Arithmetic Right (SAR)



SAL and SAR Instructions

- SAL: Shift Arithmetic Left is identical to SHL
- SAR: Shift Arithmetic Right
  - Performs a right arithmetic shift on the destination operand



- SAR preserves the number's sign

```
mov dl,-80    ; DL = 10110000b
sar dl,1      ; DL = 11011000b = -40, CF = 0
sar dl,2      ; DL = 11110110b = -10, CF = 0
```

Your Turn . . .

Indicate the value of AL and CF after each shift

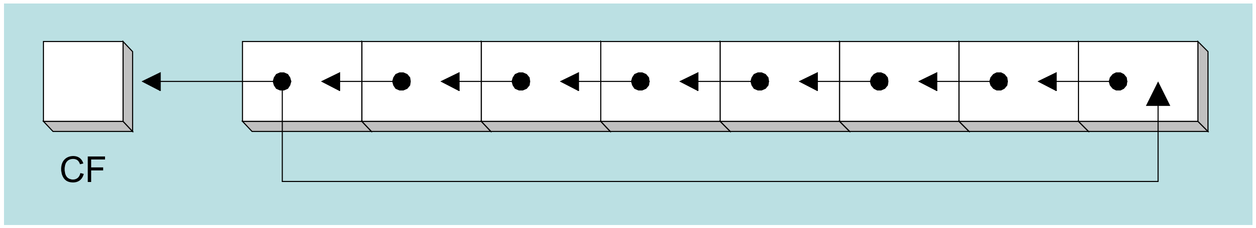
```
mov al,6Bh        ; al = 01101011b
shr al,1          ; al = 00110101b = 35h, CF = 1
shl al,3          ; al = 10101000b = A8h, CF = 1
mov al,8Ch        ; al = 10001100b
sar al,1          ; al = 11000110b = C6h, CF = 0
sar al,3          ; al = 11111000b = F8h, CF = 1
```

Effect of Shift Instructions on Flags

- The CF is the last bit shifted
- The OF is defined for single bit shift only
  - It is 1 if the sign bit changes
- The ZF, SF and PF are affected according to the result
- The AF is unaffected

ROL Instruction

- ROL is the Rotate Left instruction
  - Rotates each bit to the left, according to the count operand
  - Highest bit is copied into the Carry Flag and into the Lowest Bit
- No bits are lost



```
mov al,11110000b
rol al,1             ; AL = 11100001b, CF = 1
mov dl,3Fh           ; DL = 00111111b
rol dl,4             ; DL = 11110011b = F3h, CF = 1
```

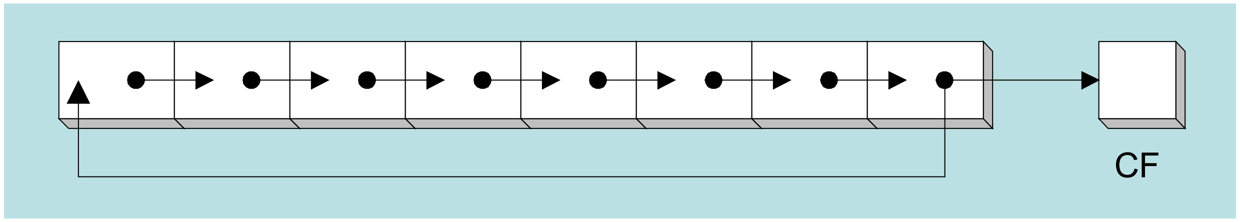
ROR Instruction

❖ ROR is the Rotate Right instruction

✧ Rotates each bit to the right, according to the count operand

✧ Lowest bit is copied into the Carry flag and into the highest bit

❖ No bits are lost



```
mov al,11110000b
ror al,1                ; AL = 01111000b, CF = 0
mov dl,3Fh              ; DL = 00111111b
ror dl,4                ; DL = F3h, CF = 1
```