SHLD Instruction

- SHLD is the Shift Left Double instruction
- Syntax: **`SHLD`** ***`destination, source, count`***
  - Shifts a *destination* operand a given *count* of bits to the left
- The rightmost bits of *destination* are filled by the leftmost bits of the *source* operand
- The *source* operand is not modified
- Operand types:

```
SHLD reg/mem16, reg16, imm8/CL
SHLD reg/mem32, reg32, imm8/CL
```

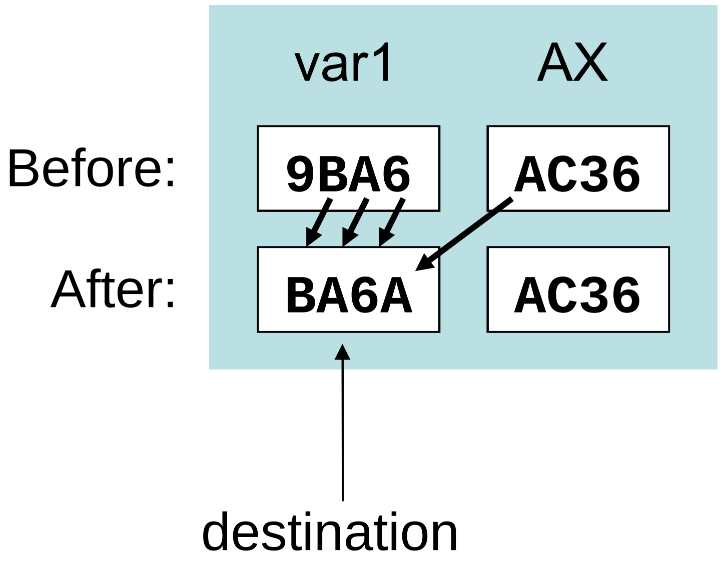
SHLD Example

Shift variable **`var1`** 4 bits to the left

Replace the lowest 4 bits of **`var1`** with the high 4 bits of AX

```
.data
var1 WORD 9BA6h
.code
mov  ax, 0AC36h
shld var1, ax, 4
```

destination source count



Only the *destination* is modified, not the *source*

SHRD Instruction

- SHRD is the Shift Right Double instruction
- Syntax: **`SHRD`** ***`destination, source, count`***
  - Shifts a *destination* operand a given *count* of bits to the right
- The leftmost bits of *destination* are filled by the rightmost bits of the *source* operand
- The *source* operand is not modified
- Operand types:

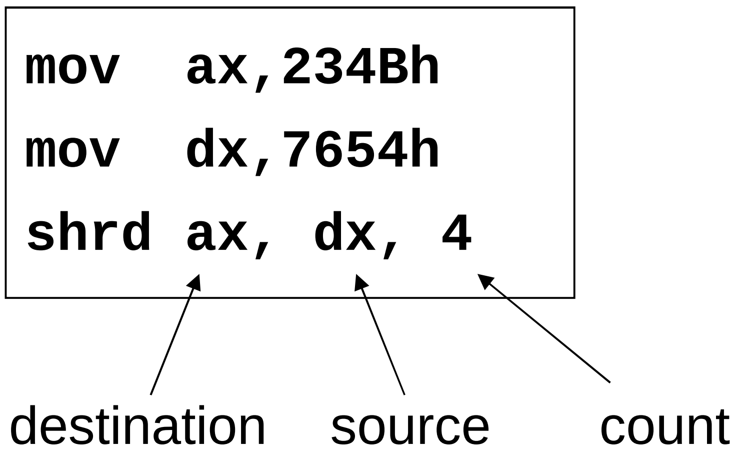
```
SHRD reg/mem16, reg16, imm8/CL
SHRD reg/mem32, reg32, imm8/CL
```

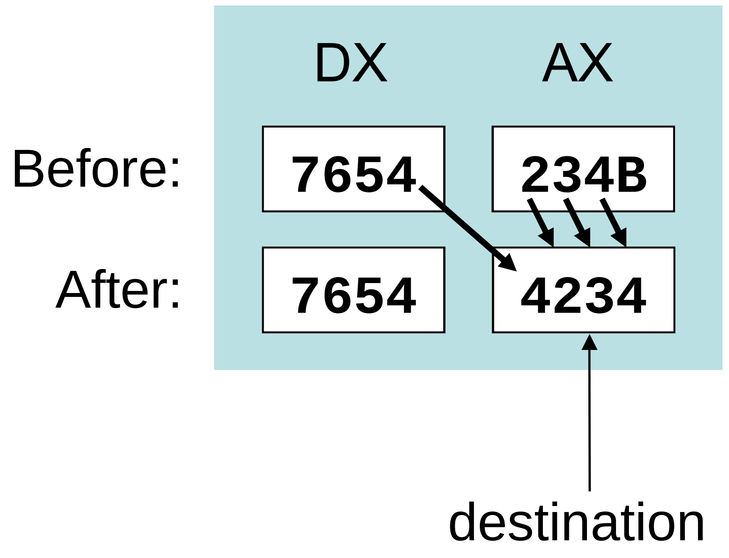
SHRD Example

Shift AX 4 bits to the right

Replace the highest 4 bits of AX with the low 4 bits of DX

```
mov  ax,234Bh
mov  dx,7654h
shrd ax, dx, 4
```





Only the *destination* is modified, not the *source*

Your Turn . . .

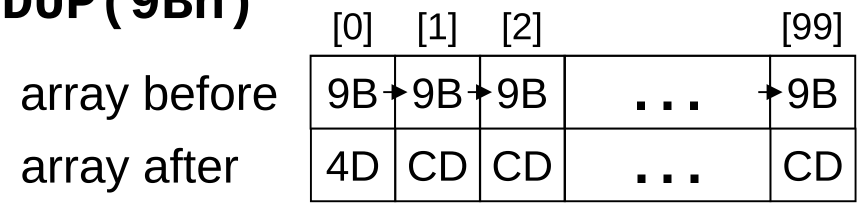
Indicate the values (in hex) of each destination operand

```
mov  ax,7C36h
mov  dx,9FA6h
shld dx,ax,4      ; DX = FA67h
shrd ax,dx,8      ; AX = 677Ch
```

Shifting Bits within an Array

- ❖ Sometimes, we need to shift all bits within an array
  - ✧ Example: moving a bitmapped image from one screen to another
- ❖ Task: shift an array of bytes 1 bit right

```
.data
   ArraySize  EQU 100
   array BYTE ArraySize DUP(9Bh)
.code
   mov ecx, ArraySize
   mov esi, 0
   clc                    ; clear carry flag
L1:
   rcr array[esi], 1      ; propagate the carry flag
   inc esi                ; does not modify carry
   loop L1                ; does not modify carry
```



Binary Multiplication

- You know that SHL performs multiplication efficiently
  - When the multiplier is a power of 2
- You can factor any binary number into powers of 2
  - Example: multiply EAX by 36
    - Factor 36 into (4 + 32) and use distributive property of multiplication
  - EAX * 36 = EAX * (4 + 32) = EAX * 4 + EAX * 32

```
mov ebx, eax           ; EBX = number
shl eax, 2             ; EAX = number * 4
shl ebx, 5             ; EBX = number * 32
add eax, ebx           ; EAX = number * 36
```

Your Turn . . .

Multiply EAX by 26, using shifting and addition instructions

Hint: 26 = 2 + 8 + 16

```
mov   ebx, eax          ; EBX = number
shl   eax, 1            ; EAX = number * 2
shl   ebx, 3            ; EBX = number * 8
add   eax, ebx          ; EAX = number * 10
shl   ebx, 1            ; EBX = number * 16
add   eax, ebx          ; EAX = number * 26
```

Multiply EAX by 31, Hint: 31 = 32 – 1

```
mov   ebx, eax          ; EBX = number
shl   eax, 5            ; EAX = number * 32
sub   eax, ebx          ; EAX = number * 31
```

Convert Number to Binary String

Task: Convert Number in EAX to an ASCII Binary String

Receives: EAX = Number
ESI = Address of binary string

Returns: String is filled with binary characters ’0’ and ’1’

```
ConvToBinStr PROC USES ecx esi
    mov  ecx,32
L1: rol  eax,1
    mov  BYTE PTR [esi],'0'
    jnc  L2
    mov  BYTE PTR [esi],'1'
L2: inc  esi
    loop L1
    mov  BYTE PTR [esi], 0
    ret
ConvToBinStr ENDP
```

Rotate left most significant bit of EAX into the Carry flag; If CF = 0, append a ’0’ character to a string; otherwise, append a ’1’; Repeat in a loop 32 times for all bits of EAX.

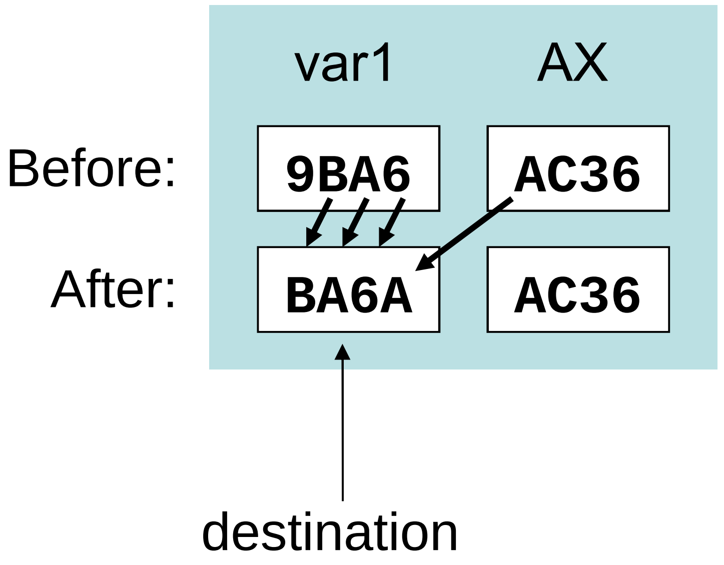
SHLD Example

Shift variable **`var1`** 4 bits to the left

Replace the lowest 4 bits of **`var1`** with the high 4 bits of AX

```
.data
var1 WORD 9BA6h
.code
mov  ax, 0AC36h
shld var1, ax, 4
```

destination   source   count



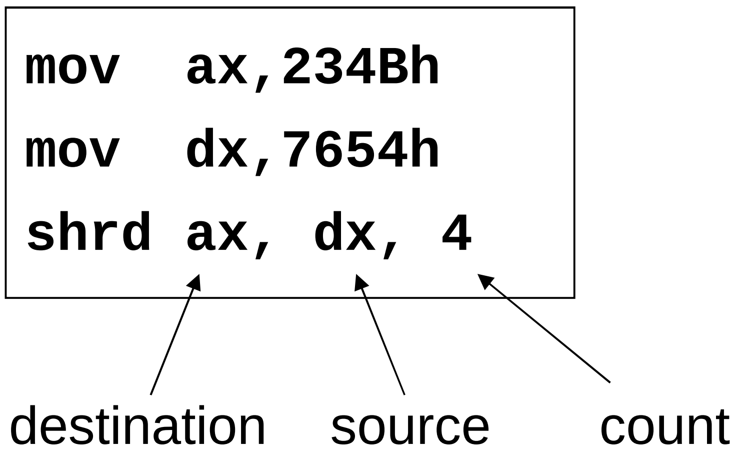
Only the *destination* is modified, not the *source*

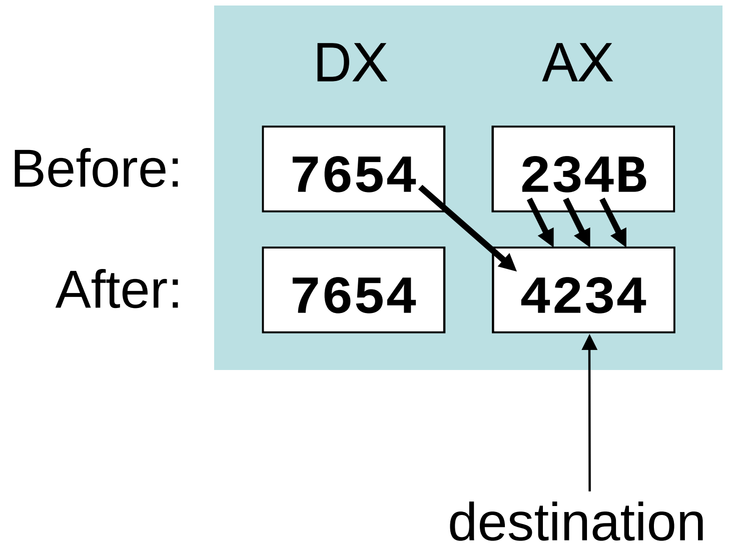
SHRD Example

Shift AX 4 bits to the right

Replace the highest 4 bits of AX with the low 4 bits of DX

```
mov  ax,234Bh
mov  dx,7654h
shrd ax, dx, 4
```





Only the *destination* is modified, not the *source*

Convert Number to Binary String

Task: Convert Number in EAX to an ASCII Binary String

Receives: EAX = Number
ESI = Address of binary string

Returns: String is filled with binary characters '0' and '1'

```
ConvToBinStr PROC USES ecx esi
    mov  ecx,32
L1: rol  eax,1
    mov  BYTE PTR [esi],'0'
    jnc  L2
    mov  BYTE PTR [esi],'1'
L2: inc  esi
    loop L1
    mov  BYTE PTR [esi], 0
    ret
ConvToBinStr ENDP
```

Rotate left most significant bit of EAX into the Carry flag; If CF = 0, append a '0' character to a string; otherwise, append a '1'; Repeat in a loop 32 times for all bits of EAX.

Convert Number to Hex String

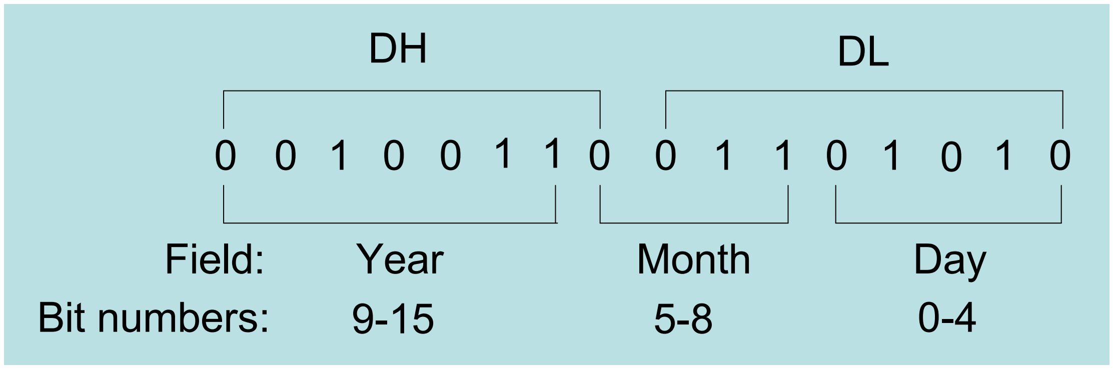
Task:	Convert EAX to a Hexadecimal String pointed by ESI
Receives:	EAX = Number, ESI= Address of hex string
Returns:	String pointed by ESI is filled with hex characters '0' to 'F'

```
ConvToHexStr PROC  USES ebx ecx esi
    mov  ecx, 8                  ; 8 iterations, why?
L1: rol  eax, 4                  ; rotate upper 4 bits
    mov  ebx, eax
    and  ebx, 0Fh                ; keep only lower 4 bits
    mov  bl,  HexChar[ebx]       ; convert to a hex char
    mov  [esi], bl               ; store hex char in string
    inc  esi
    loop L1                      ; loop 8 times
    mov  BYTE PTR [esi], 0       ; append a null byte
    ret
HexChar BYTE "0123456789ABCDEF"
ConvToHexStr ENDP
```

Isolating a Bit String

❖ MS-DOS date packs the year, month, & day into 16 bits

✧ Year is relative to 1980



In this example:

Day = 10
Month = 3
Year = 1980 + 19

Date = March 10, 1999

Isolate the Month field:

```
mov ax,dx             ; Assume DX = 16-bit MS-DOS date
shr ax,5              ; shift right 5 bits
and al,00001111b      ; clear bits 4-7
mov month,al          ; save in month variable
```