Assembly

- look at questions in Chapter 3, 4, & 5 will be on Exam
- no division on the Exam

- ;Val = sVal1 x sVal2

```
mov ax, sVal1
imul sVal2          ; dx:ax
mov iVal1, eax      mov word ptr iVal, ax      or
                    mov word ptr iVal+2, dx
```

- qVal = bVal1 x lVal2

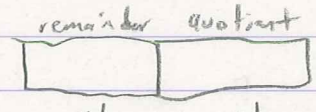
```
mov al, bVal1
cbw                 ; ax = bVal1
cwde                ; eax = bVal1
mov ebx, dword ptr lVal2    ; cannot go down
imul ebx            ; edx:eax
mov ebx, offset qVal
mov [ebx], eax
mov [ebx+4], edx
```

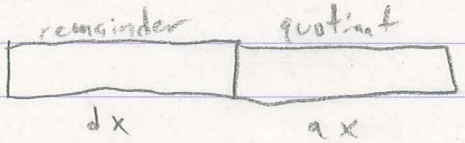
```
mov ebx, offset iVal
mov [ebx], ax
mov [ebx+2], dx
```

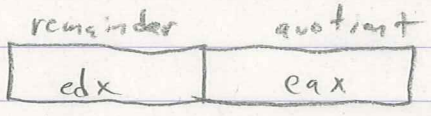
- Division - idiv (keeps sign)

```
idiv op1
```

if size of op1 = 1, then quotient is in ax register
if size of op1 = 2, then quotient should be in dx:ax
if size = 4, then should be in edx:eax







- program crashes if dividing by 0, check for this
- when multiplying, you get overflow if anything other than sign bit is there
- bT byte 7
  sX word 29

```
mov ax, sX
idiv bT          ; quotient is in al, remainder is in ah
```

remainder	quotient
01	04

- dividend must be twice as big as quotient
  2 bytes / 1 byte = 1 byte

```
sY = sX / bT
mov ax, sX
idiv bT
cbw
mov sY, ax
```

- sY = bT / sX

```
mov al, bT
cbw              ; ax = bT
cwd              ; dx:ax = bT
idiv sX          ; dx = remainder, ax = quotient
mov sY, ax
```

- ; X = sY / sX

```
mov ax, sY
cwd                 ; dx:ax
idiv sX             ; dx = remainder, ax = quotient
```

x `mov iX, dword ptr ax`

```
cwde
mov iX, eax
```

- ; X = sY / bT

```
mov ax, sY
idiv bT             ; ah = remainder, al = quotient
cbw                 ; ax = quotient
cwde                ; eax = quotient
mov iX, eax         ; iX = eax
```

- ; Y = sY % bVal

```
mov ax, sY          ; ax = sY
idiv bT             ; ah = remainder, al = quotient
mov al, ah          ; move remainder into al
cbw                 ; ax = remainder
cwde                ; eax = remainder
mov iY, eax
```

- sY = iVal / sVal;

x

```
mov eax, iVal       ; eax = iVal
idiv sVal           ; dx = remainder, ax = quotient
mov sY, ax          ; sY = ax
```

```
mov dx, word ptr iVal + 2
mov ax, word ptr iVal
```

OR

```
movsx ebx, sVal                 mov sY, ax
mov eax, iVal
cdq                 ; edx:eax = iVal
idiv ebx
```

- ; X = sVal % iVal

```
mov ax, sVal
cwde          ; eax = sVal
cdq           ; edx:eax = sVal
idiv iVal     ; eax = quotient
mov iX, edx   ; iX = remainder
```

- iVal = sVal % bVal;

```
mov  ax, sVal
idiv bVal     ; ah = remainder, al = quotient
mov al, ah    ; al = ah
cbw           ; ax
cwde          ; eax
mov iVal, eax ; iVal = eax
```

- Exam Review
- assembling, linking, loading
assembler gives object file - program in machine code
linker - adds external libraries
loaders - uses offsets to resolve actual addresses

Hollerith code for A = 12-1