Lecture # 9

```
.486                                    ; create 32 bit code
.model flat, stdcall                    ; 32 bit memory model
option casemap:none                     ; case sensitive

; -----------------------------------------------
; Irvine library
; -----------------------------------------------

include     \Irvine\Irvine32.inc
includelib  \Irvine\Irvine32.lib
includelib  \Irvine\kernel32.lib
includelib  \Irvine\user32.lib

.data
   val1     dword  10000h
   val2     dword  40000h
   val3     dword  20000h
   finalVal dword  ?

.code
main PROC
      mov       eax,val1            ; start with 10000h
      add       eax,val2            ; add 40000h
      sub       eax,val3            ; subtract 20000h
      mov       finalVal,eax        ; store the result (30000h)
      call      DumpRegs            ;
      call      WaitMsg             ; wait for a keypress
      exit
main ENDP
END main
```

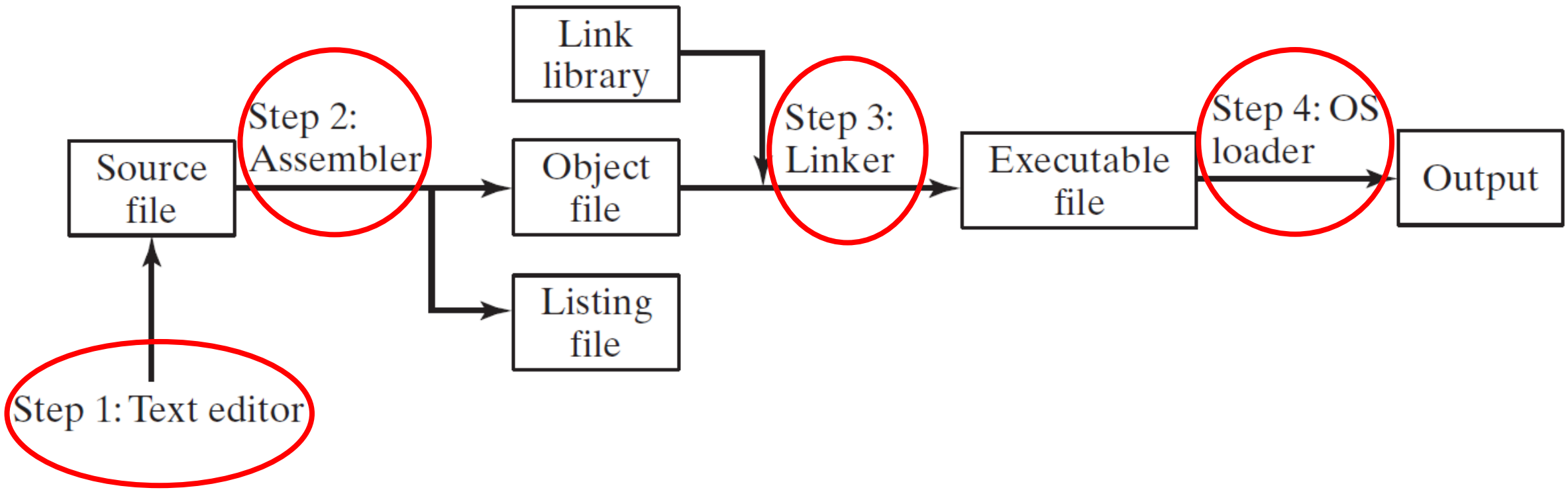
Include file

Library files

Functions from Irvine book library

Specific the name of startup function

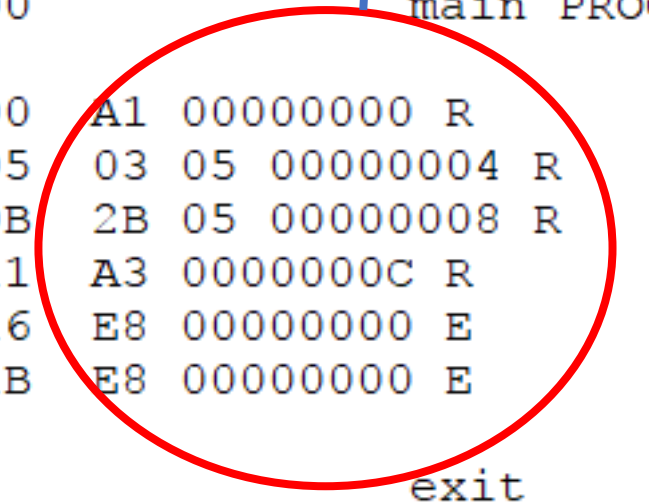
FIGURE 3–1 Assemble-Link-Execute Cycle.



Listing File (.lst)

Offsets

Machine Codes Generated by the Assembler



```
00000000                        .data
00000000 00010000                    val1     dword  10000h
00000004 00040000                    val2     dword  40000h
00000008 00020000                    val3     dword  20000h
0000000C 00000000                    finalVal dword  ?

00000000                        .code
00000000                        main PROC

00000000  A1 00000000 R              mov   eax,val1              ; start with 10000h
00000005  03 05 00000004 R           add   eax,val2              ; add 40000h
0000000B  2B 05 00000008 R           sub   eax,val3              ; subtract 20000h
00000011  A3 0000000C R              mov   finalVal,eax          ; store the result (30000h)
00000016  E8 00000000 E               call DumpRegs                ; ldsjfldsjfs
0000001B  E8 00000000 E               call  WaitMsg                ; wait for a keypress

                                     exit
00000027                             main ENDP
                                     END main
```

```
list BYTE 10,20,30,40

list BYTE 10,20,30,40
     BYTE 50,60,70,80
     BYTE 81,82,83,84

list1 BYTE 10, 32, 41h, 00100010b
list2 BYTE 0Ah, 20h, 'A', 22h

greeting1 BYTE "Good afternoon",0
greeting2 BYTE 'Good night',0
greeting1 BYTE 'G','o','o','d'....etc.
```

```
greeting1 BYTE "Welcome to the Encryption Demo program "
  BYTE "created by Kip Irvine.",0dh,0ah
  BYTE "If you wish to modify this program, please "
  BYTE "send me a copy.",0dh,0ah,0
```

```
BYTE 20 DUP(0)                  ; 20 bytes, all equal to zero
BYTE 20 DUP(?)                  ; 20 bytes, uninitialized
BYTE  4 DUP("STACK")            ; 20 bytes: "STACKSTACKSTACKSTACK"

array WORD 5 DUP(?)                ; 5 values, uninitialized
```

t a = 3,

Address Modes.

MOV AX, BX. → Register Address
MOV AX, [BX] → Regis Indir Address.

MOV AL, 10 — Immedia Address

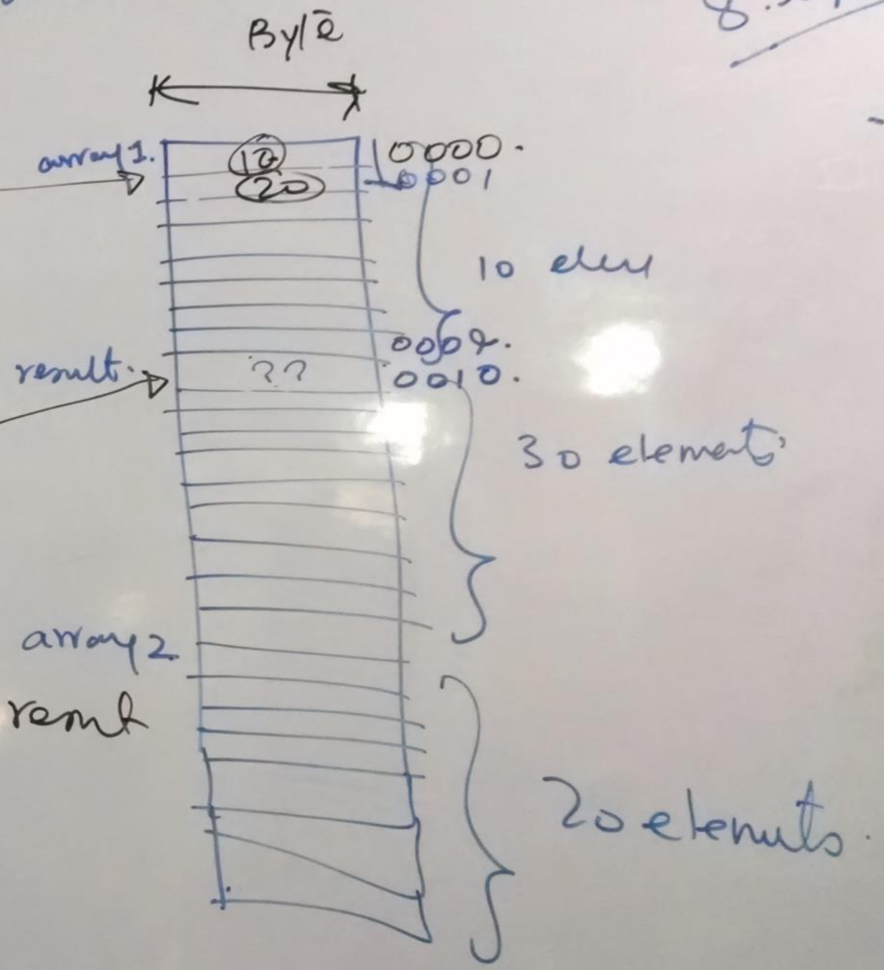
MOV AL, array1

MOV AL, array1

Data pointer

```
        MOV BX, offset array1 ; array2.
        MOV SI, offset result ; result
        MOV CX, 0.
HERE:   MOV AX, [BX]
        MOV [SI], AX
        INC BX.
        INC SI
        INC CX.
        CMP CX, 10. ; 10 times.
        JL. HERE.
```

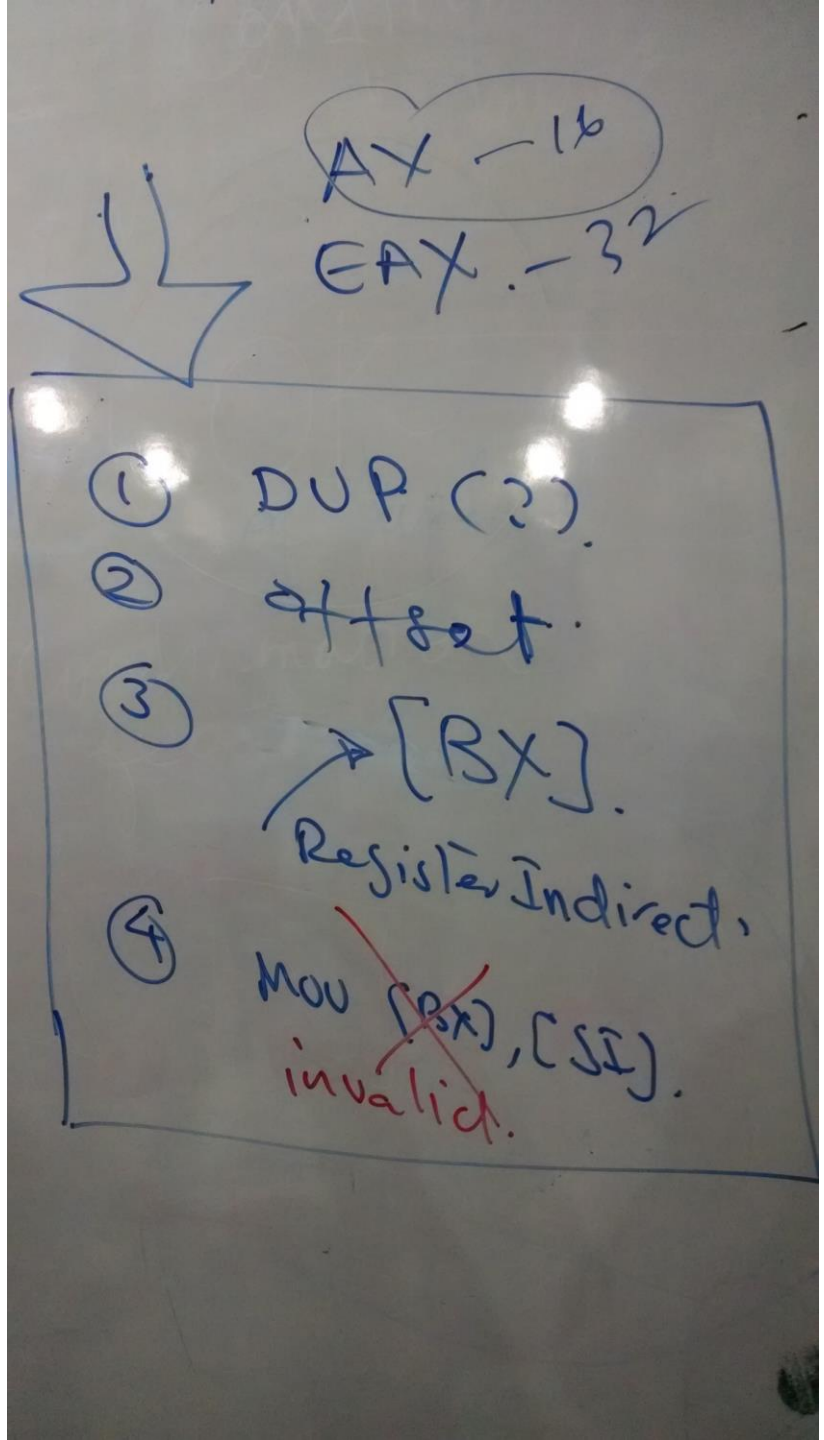
BX = 0000
SI = 0010



8:33 AM

.data

.code


AX - 16
EAX - 32
1 DUP (?).
2 ~~offset~~.
3 [BX].
Register Indirect.
4 MOV [BX], [SI].
invalid.

Must see slides
Topics not covered in class at not part of syllabus

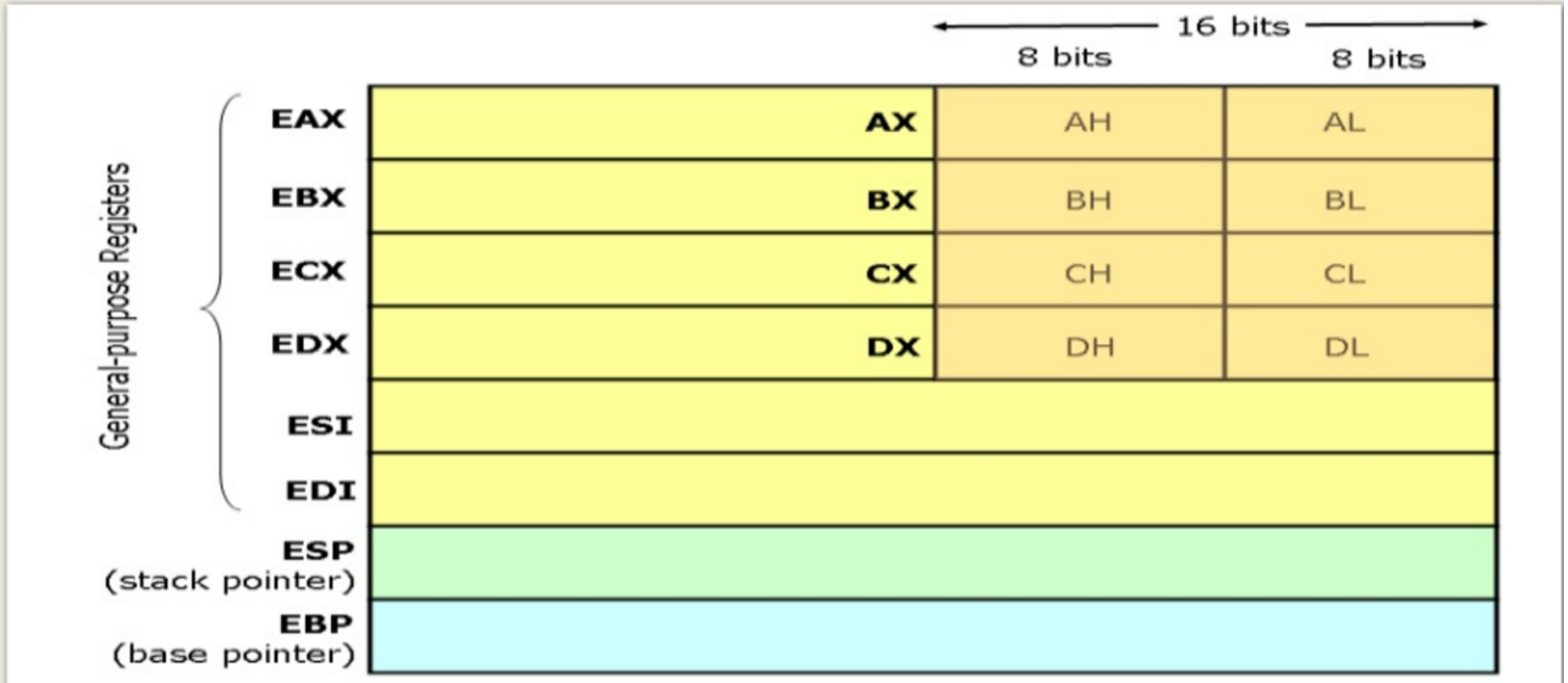
x86 Lineage

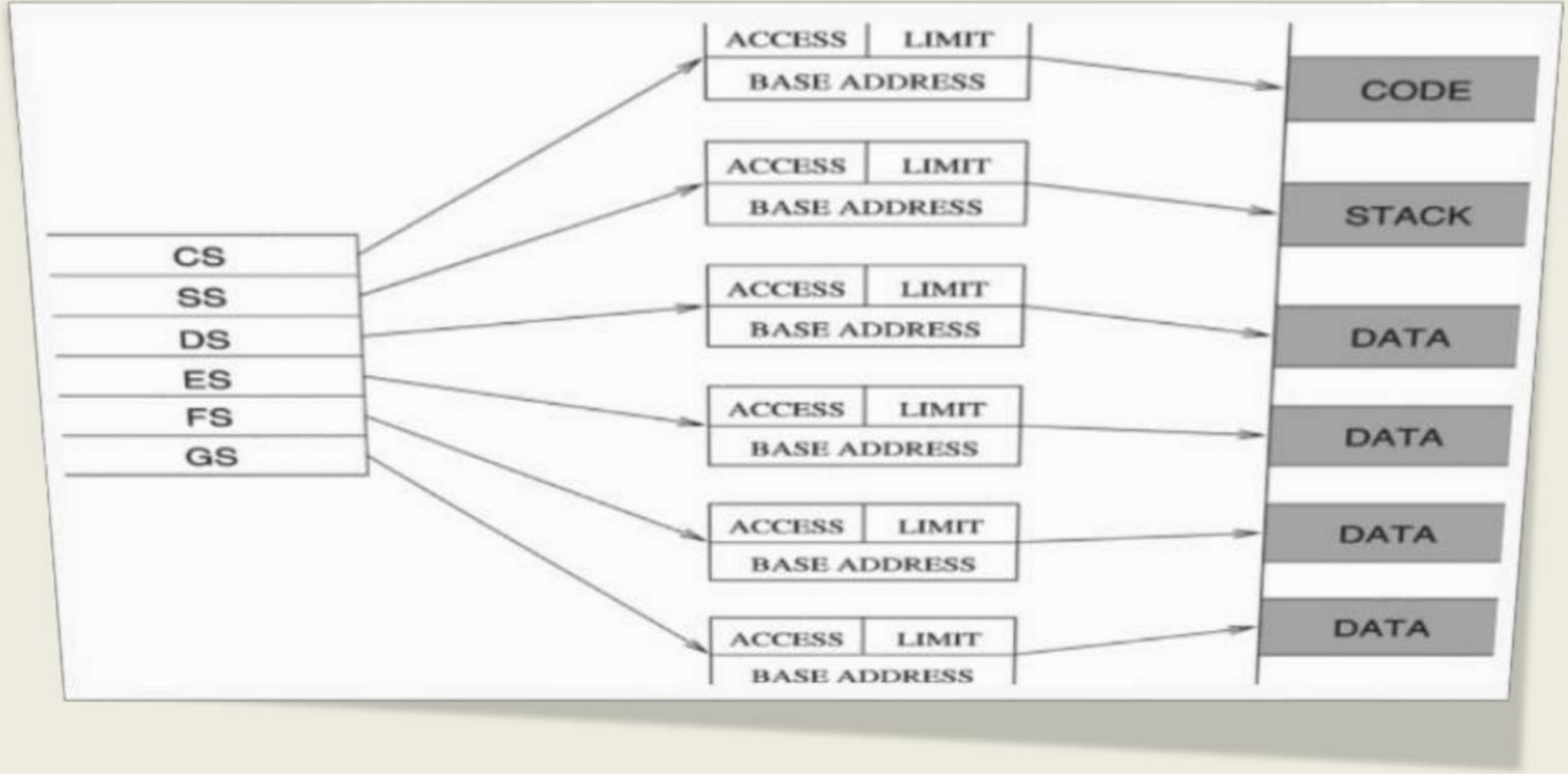
The x86 architecture, which has been enhanced numerous times, comes from the Intel 8088, the CPU in the first IBM PC in 1981. The 8088 was a slower version of the 8086, which begat the 80186, 286, 386, 486, Pentium and subsequent CPU families.

x86 PROCESSORS (from Intel)

Bits	Family	Clock Speeds (approximate range)	Bus Size (bits)	Max RAM	Floppy Disk	Hard Disk Range	OS
64	Core i3, i5, i7	2.6 - 3.3GHz	64	64GB	3.5" 1.44MB	30GB-2TB	Win7 Win Vista Win XP Win 2000 Win NT Win 95/98 Win 3.x Linux SCO Unix Solaris DOS DR DOS OS/2 Misc DOS Multiuser
	Core 2 Duo	1.8 - 2.6GHz					
	Pentium 4	3 - 3.8GHz					
	Xeon	2.2 - 3.6GHz					
	Pentium D	2.8 - 3.4GHz					
32	Core Duo	1.6 - 2.2GHz					
	Pentium 4	1.4 - 2.8GHz					
	Xeon	400MHz - 3.2GHz					
	Celeron	266MHz - 2.4GHz		4GB	3.5" 1.44MB	500MB-60GB	
	Pentium III	450MHz - 1.2GHz					
	Pentium II	233 - 450MHz					
	Pentium Pro	150 - 233MHz		64GB			
	Pentium	60 - 200MHz		4GB			
	486DX	25 - 100MHz	32		5.25" 1.2MB	200 - 500MB	
	486SX	20 - 40MHz					
	386DX	16 - 40MHz				60 - 200MB	
	386SX	16 - 33MHz					
	386SL	20 - 25MHz					
16	286	6 - 12MHz	16	16MB	5.25" 1.2MB	20-80MB	DOS DR DOS Win 3.x OS/2 1.x
	8086	5 - 10MHz	8	1MB	5.25" 360KB	10-20MB	DOS DR DOS
	8088	5MHz					

GPRS(General Purpose Registers):





Segment Registers

"Segmentation provides a mechanism for dividing the processor's addressable memory space (called the **linear address space**) into smaller protected address spaces called **segments**

8086 Addressing Modes

Immediate - The data is provided in the instruction.

Eg.

1. MOV BL, 26H ;Copies 8-bit data 26H into BL register
2. MOV CX, 4567H ;Copies 16-bit data 4567H into CX register pair

Direct - The instruction operand specifies the memory address (offset) where data is located.

Eg.

1. MOV CL, [9823H] ;9823H is the effective address [EA] directly written in the instruction

Register - References the data is in a register or in a register pair.

Eg.

1. MOV BX, CX ;Copies the 16-bit contents of CX into BX
2. MOV CL, BL ; Copies 8-bit contents of BL into CL.

Register Indirect - Instruction specifies a register containing an address, where data is located. This addressing mode works with SI, DI, BX and BP registers.

Eg. GIVEN INFO: [DI]=0030H, DS=7205H

1. MOV [DI], BX ;Here, Effective Address (EA)=[DI]
   ;Physical Address (PA)=10H*DS+EA→72050H+0030H=72080H
   ;Contents of BX pair is copied to 72080H & 72081H resp.