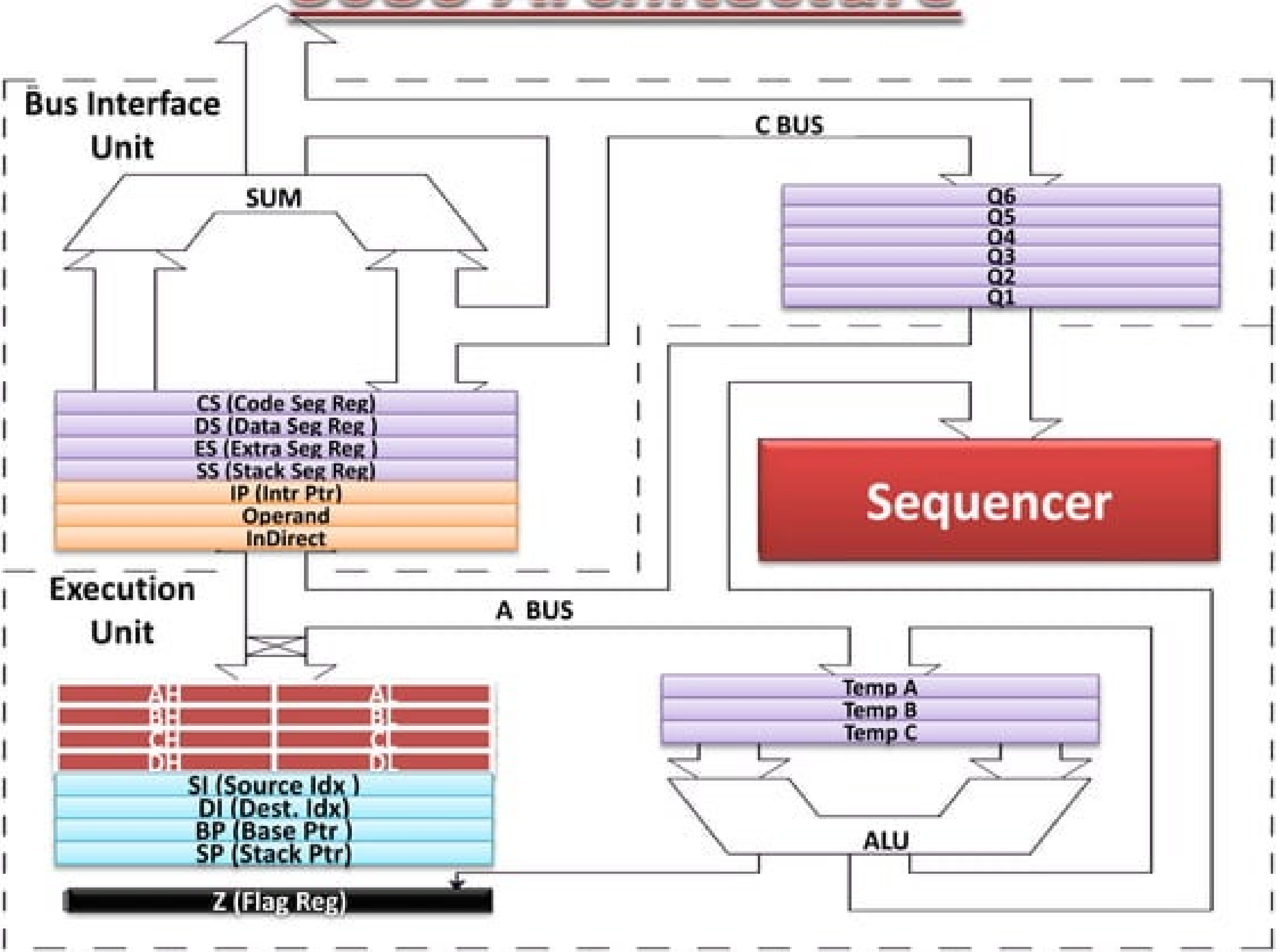
Detail of x86 Assembly Language Programming

Dr A Sahu
Dept of Computer Science & Engineering
IIT Guwahati

Outline

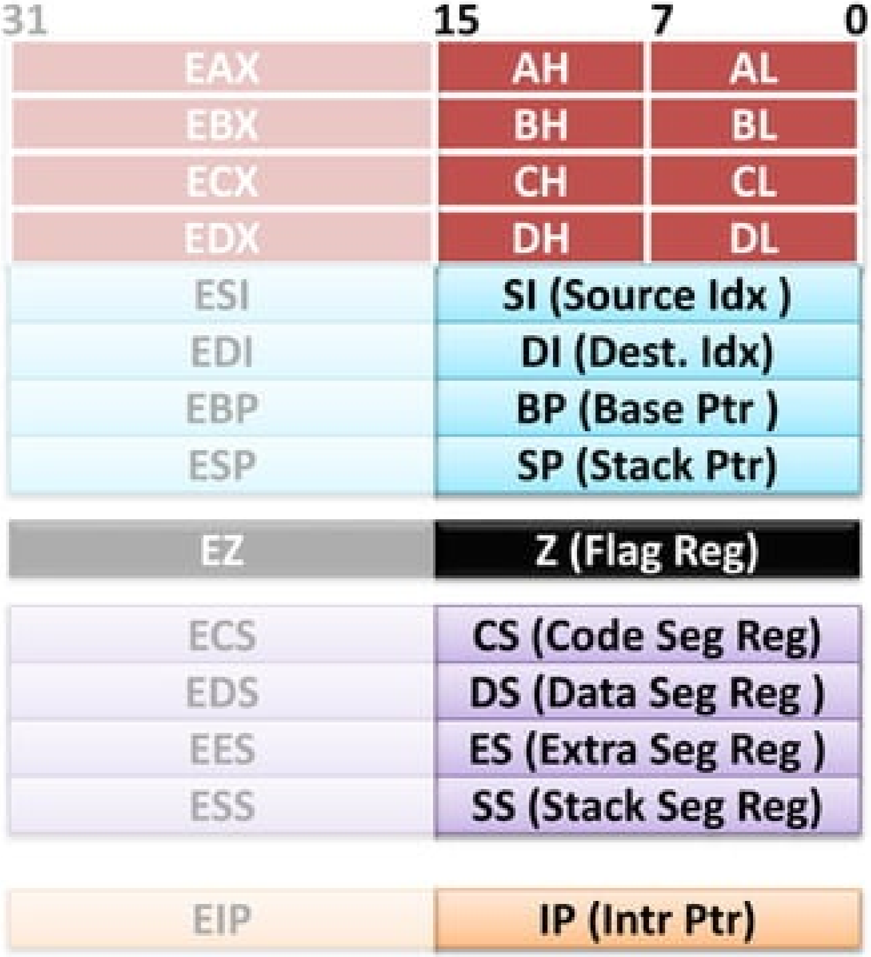
- 8086
  - Block diagram (Data Path), Registers
- Memory Model
  - Stack, Data and Code Segment
- Instruction Set of x86
- Addressing mode
- Procedure and subroutine
- Examples programs in C/C++ assembly
- Peripheral device and Assembly program

8086 Architecture



8086 & x86 Registers

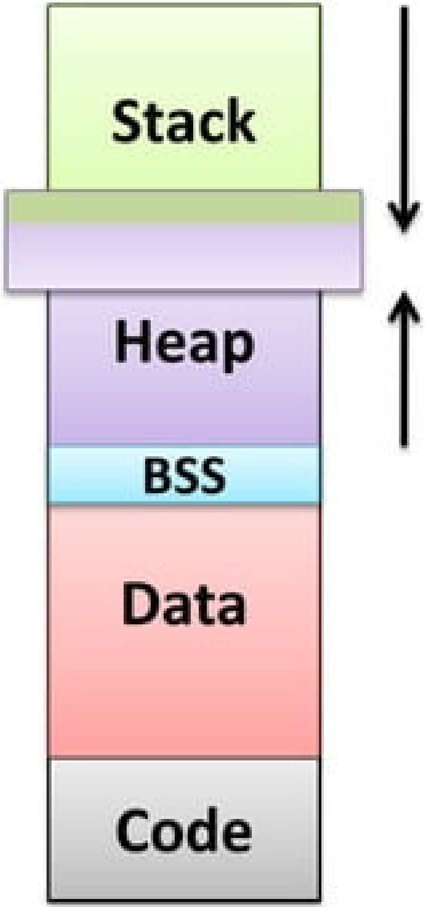
- **AX** - accumulator reg
- **BX** - base address reg
- **CX** - count reg
- **DX** - data reg
- **SI** - source index reg
- **DI** - dest index reg
- **BP** - base pointer.
- **SP** - stack pointer.



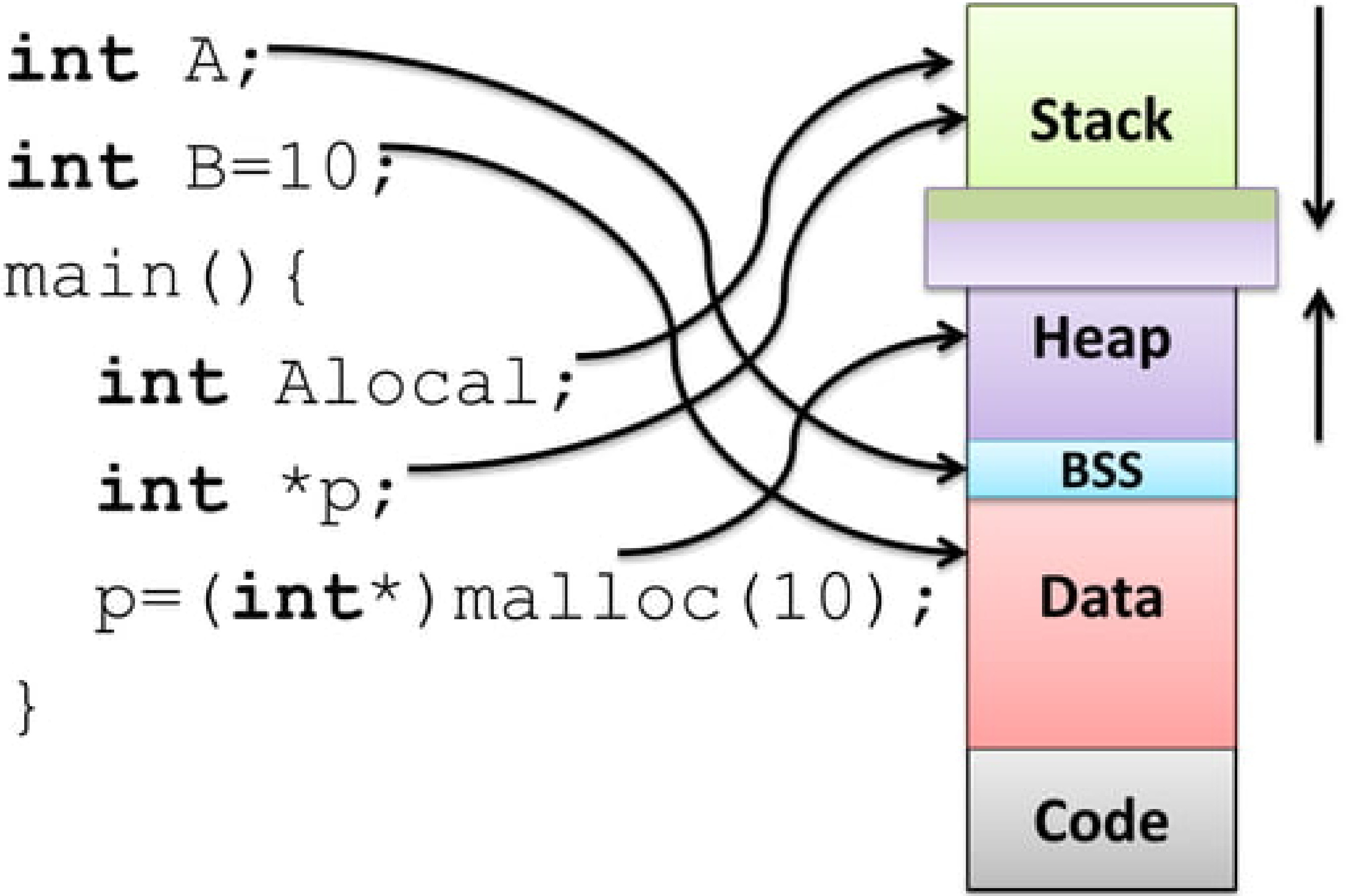
31	15	7 0
EAX	AH	AL
EBX	BH	BL
ECX	CH	CL
EDX	DH	DL
ESI	SI (Source Idx)	
EDI	DI (Dest. Idx)	
EBP	BP (Base Ptr)	
ESP	SP (Stack Ptr)	
EZ	Z (Flag Reg)	
ECS	CS (Code Seg Reg)	
EDS	DS (Data Seg Reg)	
EES	ES (Extra Seg Reg)	
ESS	SS (Stack Seg Reg)	
EIP	IP (Intr Ptr)	

Memory layout of C program

- Stack
  - automatic (default), local
  - Initialized/uninitialized
- Data
  - Global, static, extern
  - BSS: Block Started by Symbol
  - BBS: Uninitialized Data Seg.
- Code
  - program instructions
- Heap
  - malloc, calloc



Memory layout of C program



```
int A;
int B=10;
main(){
  int Alocal;
  int *p;
  p=(int*)malloc(10);
}
```

MASM : Hello world

```
.model small
.stack 100h  ; reserve 256 bytes of stack space
.data
  message   db  "Hello world, I'm learning Assembly$”
.code
main   proc
        mov   ax, seg message ; ax<-data seg. start addr.
        mov   ds, ax ; Initialize Seg Reg
        mov   ah, 09    ; 9 in the AH reg indicates Procedure
                        ;hould write a bit-string to the screen.
        lea   dx, message     ;Load Eff Address
        int   21h
        mov   ax,4c00h  ;  Halt for  DOS routine (Exit Program)
        int   21h
        main   endp
end main
```

Memory Model: Segment Definition

- .model small
  - Most widely used memory model.
  - The code must fit in 64k.
  - The data must fit in 64k.
- .model medium
  - The code can exceed 64k.
  - The data must fit in 64k.
- .model compact
  - The code must fit in 64k.
  - The data can exceed 64k.
- .medium and .compact are opposites.

How to define a segment

```
hellodat  SEGMENT BYTE 'DATA' ;Define the data segment
dos_pr    EQU 9           ;define a constant via EQU
strng     DB  'Hello World',13,10,'$‘; Define char string
hellodat  ENDS
```

```
hellodat  SEGMENT      ;define a segment
dos_print EQU 9        ;define a constant
strng     DB  'Hello World',13,10,'$'  ;Define char string
hellodat  ENDS
```

```
.data
dos_print EQU 9    ;define a constant
strng     DB  'Hello World',13,10,'$'  ;Define char string
```

Data Allocation Directives

- db : define byte dw: def. word (2 bytes)
- dd: def double word (4) dq : def quad word (8)
- equ : equate assign numeric expr to a name

```
.data

db  A 100 dup (?)  ;   define 100 bytes, with no initial
   values for bytes

db  “Hello”    ; define 5 bytes, ASCII equivalent of “Hello”.

dd  PtrArray 4 dup (?)      ;array[0..3] of dword

maxint equ 32767  ; define maxint=32767

count  equ    10 * 20      ; calculate a value (200)
```

MASM: Loop

- Assemby code: Loop
  - Loop simply decreases CX and checks if CX != 0, if so, a Jump to the specified memory location

```
            MOV CX,100
_LABEL:     INC AX
            LOOP  _LABEL
```

  - LOOPNZ : LOOPs when the zero flag is not set

```
            MOV CX,10
_CMPLOOP:   DEC AX
            CMP AX,3
            LOOPNE CMPLOOP
```

MASM: Nested Loop

- Assemby code: Nested Loop: One CX register

```
        mov    cx, 8
Loop1: push    cx
        mov    cx, 4
Loop2:  stmts
         loop   Loop2
         pop    cx
         stmts
         loop   Loop1
```

Operations

- Arithmetic
  - ADD, SUB, MUL, DIV
  - ADD AX, 5 *AX = 0003* → *AX = 0008*
- Logic
  - AND, OR, XOR, NOT
  - AND CH, DL *CH = 11111111 DL = 00000010* → *CH= 00000010*
- Bit manipulation
  - SHL/SHR
  - SHL AL, 1 *AL= 101101010* → *01101010 ;(SHL by 1)*
- Comparisons and jumps
  - JMP, CMP, Jxx, CALL, RET

How to evaluate expression

W = X + Y * Z

```
        mov     ax, y       ;Must compute Y * Z first since
        imul    z           ; multiplication has a higher
        add    ax, x        ; precedence than addition.
        mov     w, ax
```

Addressing in x86

- Register : MOV AX, BX ; **AX ← BX**
- Immediate : MOV AX, 3CH ; **AX← 3CH**
- Direct : MOV [2000], AX ; **0(DSx10h+2000)← AX**
- Reg indirect:MOV [BX], AX ; **0(DSx10h+BX)←AX**

- Base+Indx:

  MOV [BX+SI], AX ;**0(DSx10h+BX+SI)←AX**
- RegRelative:

  MOV [BX+4], AX ;**0(DSx10h+BX+4)←AX**
- Base Relative + Index

  MOV ARRAY[BX+SI], AX ;**0(DSx10h+ARRAY+BX+SI)←AX**
- Scaled index

  MOV [BX+2 x SI], AX ; **0(DSx10h+BX x 2+SI) ←AX**

Memory addressing

- Memory address written as
  - SEGMENT:OFFSET
  - Dereference offset with square brackets CS:[C494]
- DS is implicit: [1337] is same as DS:[1337]

DOS Interrupt 21H

- Input a single char from KBD and echo
  - Registers used: AH = 1, AL = the character inputted from keyboard.
  - Ex:
    ```
    MOV AH,1
    INT 21H
    ```
- Outputs a string of data, terminated by a $
  - Registers used: AH = 9, DX = the offset address of the data to be displayed.
  - Ex:
    ```
    MOV AH,09
    MOV DX,OFFSET MESS1
    INT 21H
    ```
- Terminates a process
  - Registers used: AH = 4CH, AL = binary return code.
  - Ex:
    ```
    MOV AH,4CH
    INT 21H
    ```

BIOS Interrupt 10H

- Option 0H – Sets video mode.
  - Registers used:

    AH = 0H, AL = Video Mode. 7H/3H – Col/BW 80X25
  - Ex:

    ```
    MOV AH, 0
    MOV AL,7
    INT 10H
    ```
- Option 2H – Sets the cursor to a specific location.
  - Registers used:

    AH = 2H, BH = 0H, DH = Row pos, DL = Col pos
  - Ex:

    ```
    MOV AH,2
    MOV BH,0
    MOV DH,12
    MOV DL,39
    INT 10H
    ```

GetChar, PutChar

- putchar(‘a`) ;

```
mov dl, ‘a‘       ;dl = ‘a‘
mov ah, 2h        ;character output subprogram
int 21h           ; call ms-dos output character
```

- c = getchar() ;

```
mov ah, 1h        ; keyboard input subprogram
int 21h           ; char input,  char is stored in al
mov c, al         ; copy character from al to c
```

Procedures

```
.model small
.stack     100h  ; reserve 256 bytes of stack space
.data
.code
main   proc
           call print40Dot
           mov   ax,4c00h  ;  Halt for  DOS routine (Exit Program)
           int   21h
           main   endp
end main
PrintSpaces    proc    near ;  print 40H dots
           mov    al, ' . '
           mov    cx, 40
           mov ah, 2h
PSLoop:  int 21H
           loop   PSLoop
           ret
PrintSpaces    endp
```

Macros

- MACRONAME MACRO {ARG}
- Examples

```
MOV_ASCII MACRO NUM, SRC,
 DST
          MOV CX, NUM
          LEA SI, SRC
          LEA DI, DST
          REP  MOVSB
          ENDM
```

- Call macro and expand
  - MOV_ASCII 5, 3320H, 4560H;
  - MOV_ASCII 50H, 1000H, 2000H;

```
MOV CX, 05
LEA SI,3320H
LEA DI, 4560H
REP MOVSB
MOV CX, 50H
LEA SI,1000H
LEA DI, 2000H
REP MOVSB
```

Macros

- MACRONAME MACRO {ARG}
- Examples

```
ADDITION MACRO X, Y, Z
        PUSH AX
        MOV AX,X
        ADD AX,Y
        MOV Z,AX
        POP AX
        ENDM
```

- Call macro and expand
  - ADDITION A1, A2, A3

```
PUSH AX
MOV AX,A1
ADD AX,A2
MOV A3,AX
POP AX
```

Summing first N integer

```
.model small
.data
        N EQU X
.code
main   proc
        mov    bx, N
        call   SUM_OF_N
        mov    ax,4c00H
        int    21h
        main   endp
end main
```

```
SUM_OF_N    proc   near
        cmp    bx, 00
        jz     BX_O
        push   bx
        dec    bx
        call   SUM_OF_N
        pop    bx
BX_O:   add    ax, bx
        ret
endp
```

Nested procedure: funny nature

```
OutsideProc    proc   near
          jmp    EndofOutside

InsideProc     proc   near
          mov    ax, 0
          ret
InsideProc     endp

EndofOutside:  call   InsideProc
          mov    bx, 0
          ret
OutsideProc    endp
```

Display strings

```
char string[]=“My ASM string display”;
void main(){
      Display (mystring) ;
}
```

```
void Display(char *string_addr[]) {
_asm{
            mov  bx, string_addr
            mov  ah,2                  ; set DOS function 2
     top :  mov dl, [bx]               ; display string
             inc bx
             cmp dl, 0
             je bottom
             int 21h
             jmp top
  bottom: mov dl,13          ;display clrf
             int 21h
             mov dl,10
             int 21h
 }
}
```

Display a base 10 number

```
void DisplayN(int  N)  {
_asm  {
            mov       ax, N
            mov       bx,10
             push     bx
        L1: mov       dx,0
            div       bx
            push      dx
            cmp       ax,0
            jnz  L1
        L2: pop       dx
            cmp       dl,10
            je    L3
            mov       ah,2
            add       dl,30h
            int       21h
            jmp  L2
       L3: mov        dl, ‘  ‘
            int       21h
 }
}
```

Reference

- Putc & Getc: Assembly Program:
  - http://www.csi.ucd.ie/staff/jcarthy/home/FirstScience.html
- W Tribel, A Singh, "*The 8086/8088 Microprocessor*", Pearson education india, 2nd, 2008
  - Macros & Routine
- Brey B B, "*The Intel Microprocessor*", Prentice Hall India, 2005
  - ASM inside C program
  - Addressing mode

Assignment 2

- Write and execute 8086 assembly language program to find value of SUM of square of first N number (for N=10, $S=1^2+2^2+3^2+4^2+..10^2$)
- Deadline: 21th Aug 2010, 11.55Mid night
- After deadline grading: Max 5 out of 10
- Send TXT version of program with file name RollNo.txt to asahu@iitg.ernet.in with Assignment one as subject of email
- Don't submit copied one: will get Negative marks

Next class Agenda

- Basic characteristics of peripheral devices
- Pin configurations of IO port
- Block device, Char device, Stream device
- Interrupts & ISR
- Mapping memory address to IO
- ………………

Thanks