Computer Structure and Language

The 8086/8088 Assembly Language

Hamid Sarbazi-Azad
Department of Computer Engineering
Sharif University of Technology (SUT)
Tehran, Iran

Example 1: Write a program to add two 10-byte integers. Translate you program to machine code.

```
Address     Machine Code                Source Code

                                        dataseg   segment
0000        ????????????????????       int1:     db      10 dup (?)
000A        ????????????????????       int2:     db      10 dup (?)
0014        ????????????????????       sum:      db      10 dup (?)
001E                                    dataseg   ends

                                        codeseg   segment
0000                                              assume  cs: codeseg, ds: dataseg;
0000  B8????      = 10111000 ???...???  start:    mov     ax,dataseg
0003  8ED8        = 10001110 11011000             mov     ds,ax
0005  BB0000      = 10111011 000...000            mov     bx,0
0008  B90A00      = 10111001 00001010 0000 0000   mov     cx,10
000B  F8          = 11111000                      clc
000C  8A4700      = 10001010 01000111 0000 0000 addnext: mov  al,int1[bx]
000F  12470A      = 00010010 01000111 0000 1010   adc     al,int2[bx]
0012  884714      = 10001000 01000111 0001 0100   mov     sum[bx],al
0015  43          = 01000011                      inc     bx
0016  E2F4        = 11101111 11110100             loop    addnext

0018  B8004C      = 10111000 0000 0000 0100 1100  mov     ax,4c00h
001B  CD21        = 11001101 0010 0001            int     21h
001D                                    codeseg   ends

                                                  end     start
```

Example 2: Write a program to (gnome) sort an array of 100 words. Use one segment only.

```
myseg   segment
        assume  cs: myseg, ds: myseg;
        dw      8000h
array:  dw      100 dup (?)

start:  mov     ax,cs       ; initialize DS
        mov     ds,ax
        mov     bx,0
next:
        mov     ax,array[bx]
        cmp     ax,array+2[bx]
        jle     continue    == jnh
        xchg    ax,array+2[bx]
        mov     array[bx],ax
        sub     bx,4
continue:
        add     bx,2
        cmp     bx,198
        jl      next

        mov     ax,4c00h    ; return to OS
        int     21h
myseg   ends
        end     start
```


Example 3: Write a program to add two 10x10 matrices using intra-segment procedure call.

```
dseg    segment
 A:     dw      10 dup (10 dup (?))
 B:     dw      10 dup (10 dup (?))
 C:     dw      10 dup (10 dup (?))
dseg    ends

sseg    segment
 words: dw      100 dup(?)
sseg    ends

cseg    segment

        assume  cs:cseg,ds:dseg,ss:sseg;

addmat  proc    near

        push    bp      ;
        push    cx      ; save
        push    bx      ; used
        push    ax      ; registers
        push    si      ;
        push    di      ;
        mov     bp,sp
        mov     cx,word ptr [bp+14]   ; N
        mov     si, word ptr [bp+16]  ; A
        mov     bx, word ptr [bp+18]  ; B
        mov     di, word ptr [bp+20]  ; C

adder:  mov     ax,word ptr [si]
        add     ax,word ptr [bx]
        mov     word ptr [di], ax
        add     si,2
        add     di,2
        add     bx,2
        loop    adder

        pop     di
        pop     si
        pop     ax
        pop     bx
        pop     cx
        pop     bp

        ret     8
addmat  endp
```

Example 3: Write a program to add two 10x10 matrices using procedure external. (cont.)

```
start:  mov     ax,dseg   ;  ==  mov ax, seg B
        mov     ds,ax
        mov     ax,sseg
        mov     ss,ax
        lea     sp,words+200   ;  ==  mov sp, offset words+200
        mov     ax, offset C
        push    ax
        mov     ax, offset B
        push    ax
        mov     ax, offset A
        push    ax
        mov     ax, 100
        push    ax

        call    near ptr addmat

        mov     ax,4c00h    ; return to OS
        int     21h

cseg    ends

        end     start
```


Macro Instruction definition

```
%*DEFINE (Macro name (parameter list))
(
 …..
      prototype code
……
)
```

Example:

```
%*define (pushr (a,b,c))
(       push a
        push b
        push c
)

%*define (popr (a,b,c))
(       pop a
        pop b
        pop c
)
```

Then we can use it as

```
%pushr (si,bx,cx)
%popr (bp,dx,ax)
```

That generate the following lines

```
        push si
        push bx
        push cx

        pop bp
        pop dx
        pop ax
```

Example 4: Write a program for mirroring a string using inter-segment procedure call.

```
seg1    segment
 str1:  dw      100 dup (?)
 str2:  dw      100 dup (?)
seg1    ends

seg2    segment
 words: dw      100 dup(?)
seg2    ends

seg3    segment

mirror  proc    far

        %pushr  (bp,cx,ax)
        %pushr  (di,si,f)

        mov     bp,sp
        mov     cx,word ptr [bp+16]   ; size
        mov     si, word ptr [bp+18]  ; str1
        mov     di, word ptr [bp+20]  ; str2

next:   cld
        lodsw
        std
        stows
        loop    next

        %popr   (f,si,di)
        %popr   (ax,cx,bp)

        ret     6
mirror  endp

seg3    ends
```


Example 4: Write a program for mirroring a string using inter-segment procedure call. (cont.)

```
 seg4    segment

         assume  cs:seg4,ds:seg1,ss:seg2
start:   mov     ax,seg1
         mov     ds,ax
         mov     ax,seg2
         mov     ss,ax
         lea     sp,words+200

         mov     ax, offset str2+198
         mov     bx, offset str1
         mov     cx,100
         %pushr  (ax,bx,cx)

         call    far ptr mirror

         mov     ax,4c00h    ; return to OS
         int     21h
seg4     ends

         end     start
```

Example 5: Write a code for moving string1 to string2 using and without using string instructions.

```
; Without string instructions
……
        mov     si, offset string1
        mov     di, offset string2
        mov     cx, length string1
move:   mov     al,[si]
        mov     [di],al
        inc     si
        inc     di
        loop    move
…….
```

```
; With string instructions
……
        mov     ax,ds
        mov     es,ds
        mov     si, offset string1
        mov     di, offset string2
        mov     cx, length string1
        cld
    rep mov     string1,string2
…….
```


Example 6: Write a program to add two 20-byte packed BCD numbers.

```
dataseg         segment
  BCD1:  db     20 dup (?)
  BCD2:  db     20 dup (?)
dataseg ends

codeseg segment
        assume  cs: codeseg, ds: dataseg;
  start:  mov   ax,dataseg
        mov     ds,ax
        mov     bx,0
        mov     cx,20
        clc
        pushf
 nextdigit:
        popf
        mov     al,BCD1[bx]
        adc     al,BCD2[bx]
        daa
        mov     BCD1[bx],al
        pushf
        inc     bx
        loop    nextdigit

        mov     ax,4c00h
        int     21h
codeseg ends

        end     start
```

Example 7: Write a program to multiply two 2-digt unpacked BCD numbers.

a_1a_0 X
b_1b_0
---
c_{02} c_{01} c_{00} +
c_{12} c_{11} c_{10}
---
c_3 c_2 c_1 c_0

```
mov     al,a
mul     b
aam
mov     word ptr c0, ax
mov     al,a+1
mul     b
aam
add     al,c0+1
aaa
mov     word ptr c0+1,ax
mov     al,a
mul     b+1
aam
mov     word ptr c1,ax
mov     al,a+1
mul     b+1
aam
add     al,c1+1
aaa
mov     word ptr c1+1,ax
```

```
mov     al,c0
mov     c,al
mov     al,c0+1
add     al,c1
aaa
mov     c+1,al
mov     al,c0+2
adc     al,c1+1
aaa
mov     c+2,al
mov     al,0
adc     al,c1+2
aaa
mov     c+3,al
```


End of Slides