Microprocessing and Interfacing Lab Session 2
Intro to Assembly Language Programming

Anubhav Elhence

Quick Recap

1. Using three addressing modes (Immediate, Register, Register-Indirect), write instructions to
   - Move the value 1133 into the register AX.
   - Swap the lower and higher bytes in AX and move them into BX (If AX is pqrs, BX should be rspq)
   - Move the value in BX to the memory location at an offset of 20 (from BX)

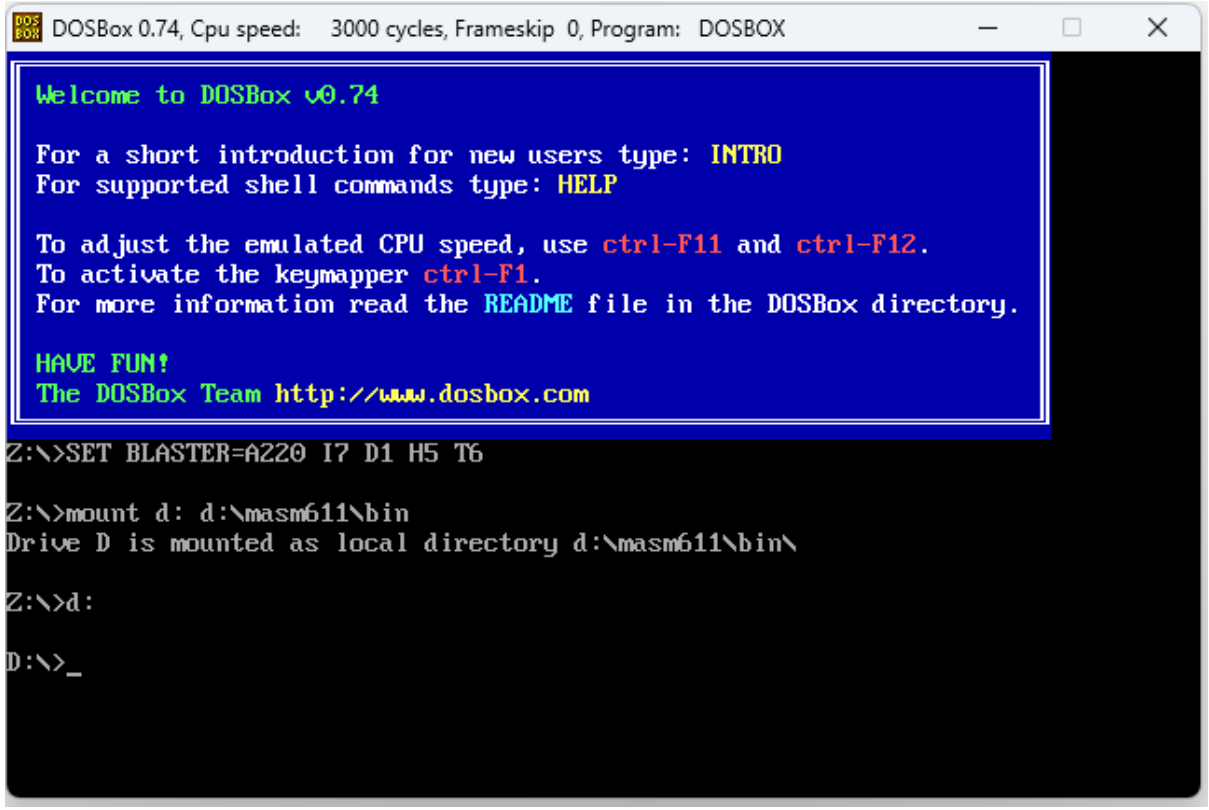
   - Note down the machine code equivalents of the four MOV statements.
   - (***Hint:*** You need to use the following commands- A to write the instructions, and U to view the machine code and unassembled instructions, T to execute and D to view the memory location)

2. Move the first letter of your name (ASCII Character) to the location DS:0120
   - (Hint: Recall the rules for the Immediate addressing mode)

3. Fill 32 (decimal) bytes of the Extra Segment with ASCII characters for the first two letters of your name. (Like "ABABAB…")
   - (***Hint:*** Use the F (Fill) command to fill a memory region with a byte pattern
   - To fill for example the first 8000h bytes of current data segment with pattern 55:
   - **F** 0 L 8000 55
   - [Syntax: **F** <start-address> L <range> <pattern>])

Follow along example

- First Let's open the DOSBOX and load masm611/bin folder AS ALWAYS



- Install VS Code (My preferred IDE) and install asm extension

- Open folder workspace in VS Code with masm611/bin and create new file ex1.asm

- Now let's write ALP for moving data in and out of the memory.
  But before that it is important to have a discussion on Segments.

Segments

- Logical Segments contain the 3 components of a program
  - code
  - data
  - stack

0859 × 10H
+ 0000H
+ FFFFH

CS
DS
ES

128K

64K

1858H

- Mapping of Logical segments to actual physical segments in memory

- MASM has assembler directives to help us manage segments

- Mostly we'll be using simplified segment directives unless there is a need to use STACK.

Segments

0000H
64K
FFFFH

- There are many models available in MASM assemb;er ranging from Tiny to Small to Huge

- To designate a model use the .MODEL statement followed by the memory model

- ex.

  .MODEL TINY

  Tiny model requires that all program and data fit into one 64K segment. HOW ?

-

```
.CODE
.DATA
.STARTUP
.EXIT
```

Continuing our Follow along example

- Create this program in VS code editor to store data into memory and do some MOV operations

ex1.asm > end

```
.model tiny
.data
2 references
data1 db 23
1 reference
data2 dw 9999h
1 reference
data3 dw 9999h
1 reference
array dw 01,02,03,04,05,06,07,08,09

.code
.startup
    mov BX,data2
    mov CX,data3
    mov data1,BL
    mov DL,data1
    mov DI,0002h
    mov AX,array[DI]
.exit
1 reference
end
```

- Let's compile and run the executable in debugx using the following commands:

a.asm | ex1.asm | DEBUG.ASM

ex1.asm > ...

```
.model tiny
.data
2 references
data1 db 23
1 reference
data2 dw 9999h
1 reference
data3 dw 9999h
1 reference
array dw 01,02,03,04,05,06,07,08,09

.code
.startup
    mov BX,data2
    mov CX,data3
    mov data1,BL
    mov DL,data1
    mov DI,0002h
    mov AX,array[DI]
.exit
1 reference
end
```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

```
The DOSBox Team http://www.dosbox.com

Z:\>SET BLASTER=A220 I7 D1 H5 T6

Z:\>mount d: d:\masm611\bin
Drive D is mounted as local directory d:\masm611\bin\

Z:\>d:

D:\>ml ex1.asm
Microsoft (R) Macro Assembler Version 6.11
Copyright (C) Microsoft Corp 1981-1993.  All rights reserved.

 Assembling: ex1.asm

Microsoft (R) Segmented Executable Linker  Version 5.31.009 Jul 13 1992
Copyright (C) Microsoft Corp 1984-1992.  All rights reserved.

Object Modules [.obj]: ex1.obj/t
Run File [ex1.com]: "ex1.com"
List File [nul.map]: NUL
Libraries [.lib]:
Definitions File [nul.def]:

D:\>_
```

- Let's see the instructions using U command

ex1.asm > ...

```
.model tiny
.data
data1 db 23
data2 dw 9999h
data3 dw 9999h
array dw 01,02,03,04,05,06,07,08,09

.code
.startup
    mov BX,data2
    mov CX,data3
    mov data1,BL
    mov DL,data1
    mov DI,0002h
    mov AX,array[DI]
.exit
end
```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUGX

```
Microsoft (R) Segmented Executable Linker  Version 5.31.009 Jul 13 1992
Copyright (C) Microsoft Corp 1984-1992.  All rights reserved.

Object Modules [.obj]: ex1.obj/t
Run File [ex1.com]: "ex1.com"
List File [nul.map]: NUL
Libraries [.lib]:
Definitions File [nul.def]:

D:\>debugx ex1.com
-u
0863:0100 8B1E1D01          MOV     BX,[011D]
0863:0104 8B0E1F01          MOV     CX,[011F]
0863:0108 881E1C01          MOV     [011C],BL
0863:010C 8A161C01          MOV     DL,[011C]
0863:0110 BF0200            MOV     DI,0002
0863:0113 8B852101          MOV     AX,[DI+0121]
0863:0117 B44C              MOV     AH,4C
0863:0119 CD21              INT     21
0863:011B 0017              ADD     [BX],DL
0863:011D 99                CWD
0863:011E 99                CWD
0863:011F 99                CWD
-
```

- But HEY WAIT, I had written data2/data3/etc but it is all replaced by some byte code.

- So how to know which memory location my data has gone?

ex1.asm > ...

```
.model tiny
.data
3 references
data1 db 23
2 references
data2 dw 9999h
2 references
data3 dw 9999h
2 references
array dw 01,02,03,04,05,06,07,08,09

.code
.startup
    lea SI, data1
    lea SI, data2
    lea SI, data3
    lea SI, array
    mov BX,data2
    mov CX,data3
    mov data1,BL
    mov DL,data1
    mov DI,0002h
    mov AX,array[DI]
.exit
1 reference
end
```

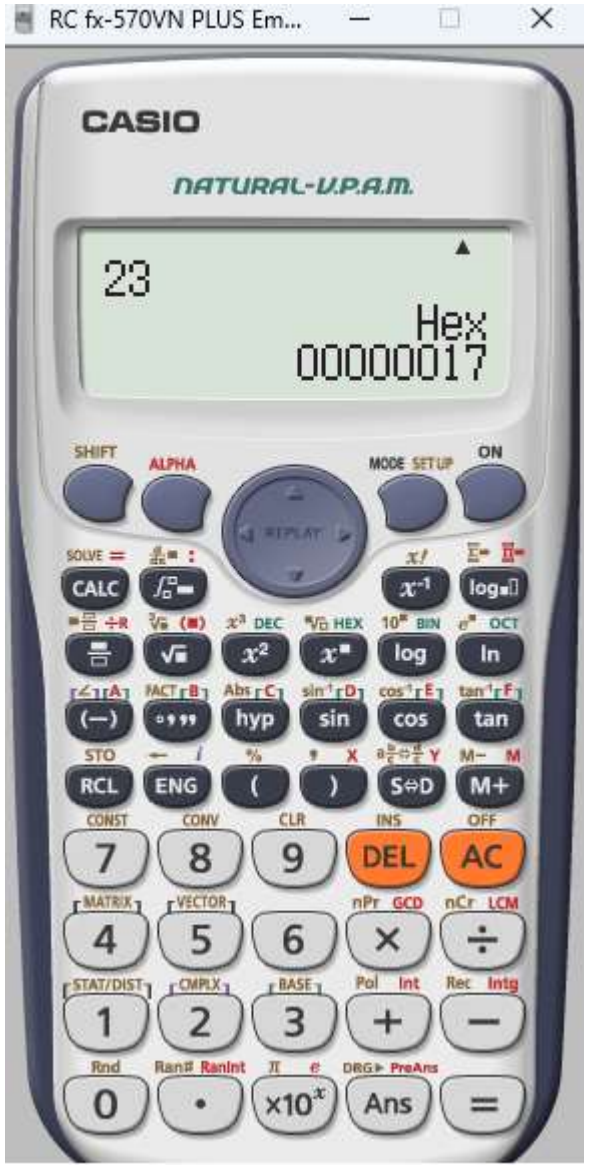
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUGX

```
-u ds:100
0863:0100 8D362C01        LEA     SI,[012C]
0863:0104 8D362D01        LEA     SI,[012D]
0863:0108 8D362F01        LEA     SI,[012F]
0863:010C 8D363101        LEA     SI,[0131]
0863:0110 8B1E2D01        MOV     BX,[012D]
0863:0114 8B0E2F01        MOV     CX,[012F]
0863:0118 881E2C01        MOV     [012C],BL
0863:011C 8A162C01        MOV     DL,[012C]
-_
```

```
-u ds:011c
0863:011C 8A162C01        MOV     DL,[012C]
0863:0120 BF0200          MOV     DI,0002
0863:0123 8B853101        MOV     AX,[DI+0131]
0863:0127 B44C            MOV     AH,4C
0863:0129 CD21            INT     21
0863:012B 0017            ADD     [BX],DL
0863:012D 99              CWD
```

- Now let's go to memory location of data1

```
-d ds:012c
0863:0120                                -           17 99 99 99              ....
0863:0130  99 01 00 02 00 03 00 04-00 05 00 06 00 07 00 08 ................
0863:0140  00 09 00 2D 14 EF 06 8B-E5 5D CB 55 8B EC 83 3E ...-.....].U...>
0863:0150  06 57 02 73 13 8A 46 08-2A E4 50 FF 76 06 9A 0E .W.s..F.*.P.v...
0863:0160  00 DB 09 83 C4 04 EB 30-8B 76 06 8A 04 22 C0 74 .......0.v...".t
0863:0170  25 2A E4 50 0E E8 F4 00-83 C4 02 23 C0 74 05 FF %*.P.......#.t..
0863:0180  46 06 EB 0D 8B 76 06 8A-04 3A 46 08 75 03 96 EB F....v...:F.u...
0863:0190  07 FF 46 06 EB D2 2B C0-5D CB 55 8B EC 83 3E 06 ..F...+.].U...>.
0863:01A0  57 02 73 15 8A 46 0A 2A-E4 50 FF 76            W.s..F.*.P.v
-d ds:012d
0863:0120                                -              99 99 99              ...
0863:0130  99 01 00 02 00 03 00 04-00 05 00 06 00 07 00 08 ................
0863:0140  00 09 00 2D 14 EF 06 8B-E5 5D CB 55 8B EC 83 3E ...-.....].U...>
0863:0150  06 57 02 73 13 8A 46 08-2A E4 50 FF 76 06 9A 0E .W.s..F.*.P.v...
0863:0160  00 DB 09 83 C4 04 EB 30-8B 76 06 8A 04 22 C0 74 .......0.v...".t
0863:0170  25 2A E4 50 0E E8 F4 00-83 C4 02 23 C0 74 05 FF %*.P.......#.t..
0863:0180  46 06 EB 0D 8B 76 06 8A-04 3A 46 08 75 03 96 EB F....v...:F.u...
0863:0190  07 FF 46 06 EB D2 2B C0-5D CB 55 8B EC 83 3E 06 ..F...+.].U...>.
0863:01A0  57 02 73 15 8A 46 0A 2A-E4 50 FF 76 08         W.s..F.*.P.v.
-
```



- Why is this 17 coming here?

- Now let's experiment with db,dw, and other data to see how they are stored in memory

ex1.asm > ...

```
.model tiny
.data
data1 db 23,'a','A', '0','*'
data2 dw 9976h,'a','A', '0','*'
data3 dw 0fefeh
array dw 01,02,03,04,05,06,07,08,09

.code
.startup
    lea SI, data1
    lea SI, data2
    lea SI, data3
    lea SI, array
    mov BX,data2
    mov CX,data3
    mov data1,BL
    mov DL,data1
    mov DI,0002h
    mov AX,array[DI]
.exit
end
```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUGX

```
List File [nul.map]: NUL
Libraries [.lib]:
Definitions File [nul.def]:

D:\>debugx ex1.com
-u
0863:0100 8D362C01          LEA     SI,[012C]
0863:0104 8D363101          LEA     SI,[0131]
0863:0108 8D363B01          LEA     SI,[013B]
0863:010C 8D363D01          LEA     SI,[013D]
0863:0110 8B1E3101          MOV     BX,[0131]
0863:0114 8B0E3B01          MOV     CX,[013B]
0863:0118 881E2C01          MOV     [012C],BL
0863:011C 8A162C01          MOV     DL,[012C]
-d ds:012c
0863:0120                           -           17 61 41 30             .aA0
0863:0130  2A 76 99 61 00 41 00 30-00 2A 00 FE FE 01 00 02 *v.a.A.0.*......
0863:0140  00 03 00 04 00 05 00 06-00 07 00 08 00 09 00 3E ...............>
0863:0150  06 57 02 73 13 8A 46 08-2A E4 50 FF 76 06 9A 0E .W.s..F.*.P.v...
0863:0160  00 DB 09 83 C4 04 EB 30-8B 76 06 8A 04 22 C0 74 .......0.v...".t
0863:0170  25 2A E4 50 0E E8 F4 00-83 C4 02 23 C0 74 05 FF %*.P.......#.t..
0863:0180  46 06 EB 0D 8B 76 06 8A-04 3A 46 08 75 03 96 EB F....v...:F.u...
0863:0190  07 FF 46 06 EB D2 2B C0-5D CB 55 8B EC 83 3E 06 ..F...+.].U...>.
0863:01A0  57 02 73 15 8A 46 0A 2A-E4 50 FF 76            W.s..F.*.P.v
-_
```

- What did you notice?

- Now let's try to get the data from a txt file

a.asm | ex1.asm | data.txt | DEBUG.ASM

BIN > data.txt

```
1   Apple*MUP
```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUGX

```
-?
DOS DebugX v1.25 help screen
assemble        A [address]
compare         C range address
dump            D [range]
dump interrupt  DI interrupt [count]
dump LDT        DL selector [count]
dump MCB chain  DM
dump ext memory DX [physical_address]
enter           E address [list]
fill            F range list
go              G [=address] [breakpts]
hex add/sub     H value1 value2
input           I[W|D] port
load program    L [address]
load sectors    L address drive sector count
move            M range address
80x86 mode      M [x] (x=0..6)
set FPU mode    MC [2|N] (2=287,N=no FPU)
set name        N [[drive:][path]progname [arglist]]
output          O[W|D] port value
proceed         P [=address] [count]
quit            Q
register        R [register [value]]
[more]
```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUGX

```
-N data.txt
-L es:0200
-d es:0200
0859:0200  41 70 70 6C 65 2A 4D 55-50 00 00 00 00 00 00 00 Apple*MUP.......
0859:0210  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
0859:0220  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
0859:0230  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
0859:0240  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
0859:0250  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
0859:0260  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
0859:0270  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ................
-_
```

Lab Task

- Create a `.txt` file that contains your First Name in small letters followed by "MUP," separated by a '*****.' For example, if your name is 'Apple,' then the content of the file should be
  - *Apple*MUP*
- Now place this string in the Extra Segment at 0200h, with the "*****" replaced by "**[**" manually and also place an additional "**]**" at the end of the string too.

ES:0200 *Apple[MUP]*

(***Hint:***
The Name command (N) is used to input a filename (actually, a file specification that can include a drive and a path) for a subsequent Load or Write command.
  - Syntax: N filespec [arguments]

The Load command (L) is used to load a file into memory. The file to load is specified with the Name command (N). The optional address parameter specifies the load address

Syntax: L [address])

ASCII Characters
- [= 5B
-] = 5D
- * = 2A

- 1. Create a .txt file containing the above string
- 2. load the value 5B into AH and 5D into AL
- 3. Load the file you created in step 1 using the L command into code segment (assume/use offset to code segment as 0200h)
- 4. Copy the values AH and AL into the appropriate locations using the MOV instruction with the appropriate offsets.)

- What can we do to figure out the hex code of "[" and "]" and "*"

a.asm | ex1.asm | data.txt | DEBUG.ASM

BIN > ex1.asm > data1

```
.model tiny
.data
3 references
data1 db '*', '[', ']'
2 references
data2 dw 9976h,'a','A', '0','*'
2 references
data3 dw 0fefeh
2 references
array dw 01,02,03,04,05,06,07,08,09

.code
.startup
    lea SI, data1
    lea SI, data2
    lea SI, data3
    lea SI, array
    mov BX,data2
    mov CX,data3
    mov data1,BL
    mov DL,data1
    mov DI,0002h
    mov AX,array[DI]
.exit
1 reference
end
```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUGX

```
List File [nul.map]: NUL
Libraries [.lib]:
Definitions File [nul.def]:

D:\>debugx ex1.com
-u
0863:0100 8D362C01          LEA     SI,[012C]
0863:0104 8D362F01          LEA     SI,[012F]
0863:0108 8D363901          LEA     SI,[0139]
0863:010C 8D363B01          LEA     SI,[013B]
0863:0110 8B1E2F01          MOV     BX,[012F]
0863:0114 8B0E3901          MOV     CX,[0139]
0863:0118 881E2C01          MOV     [012C],BL
0863:011C 8A162C01          MOV     DL,[012C]
-d 012c
0863:0120                           -           2A 5B 5D 76             *[]v
0863:0130  99 61 00 41 00 30 00 2A-00 FE FE 01 00 02 00 03 .a.A.0.*........
0863:0140  00 04 00 05 00 06 00 07-00 08 00 09 00 EC 83 3E ...............>
0863:0150  06 57 02 73 13 8A 46 08-2A E4 50 FF 76 06 9A 0E .W.s..F.*.P.v...
0863:0160  00 DB 09 83 C4 04 EB 30-8B 76 06 8A 04 22 C0 74 .......0.v...".t
0863:0170  25 2A E4 50 0E E8 F4 00-83 C4 02 23 C0 74 05 FF %*.P.......#.t..
0863:0180  46 06 EB 0D 8B 76 06 8A-04 3A 46 08 75 03 96 EB F....v...:F.u...
0863:0190  07 FF 46 06 EB D2 2B C0-5D CB 55 8B EC 83 3E 06 ..F...+.].U...>.
0863:01A0  57 02 73 15 8A 46 0A 2A-E4 50 FF 76            W.s..F.*.P.v
-_
```

To Be continued...

Thank You