Dr. Haitham Ziade
Dr. Wassim El Falou

MP2: Basic Assembly Programming (MP68000)

Ex1: Decoding Instructions:

Give the code for the following instructions:

```
MOVE.L #$2000,A1
MOVE.B (A1)+,D0
```

To test the code in the simulator enter the code directly to the memory:

```
ORG $1000
DC.W hexa code of inst1
DC.W hexa code of inst2
ORG $2000
DC.B $12,$23
END $1000
```

a) Run your program using 68000 IDE simulator (To run it first click on Project > Build all, and after click on Run > Visual Simulator)

b) You can assemble your code directly without using manual decoding. Write the following code in the simulator:

```
ORG $1000
MOVE.L #$2000,A1
MOVE.B (A1)+,D0
ORG $2000
DC.B $12,$23
END $1000
```

After building the code, examine the .lst file and .hex file (found in C:\Ide68k\Examples). You can show that assembler give the machine language code in the .lst file.

Run also the code in the simulator to show that the code do the same things in a.

Ex2: Addressing modes:

Write the following code in simulator:

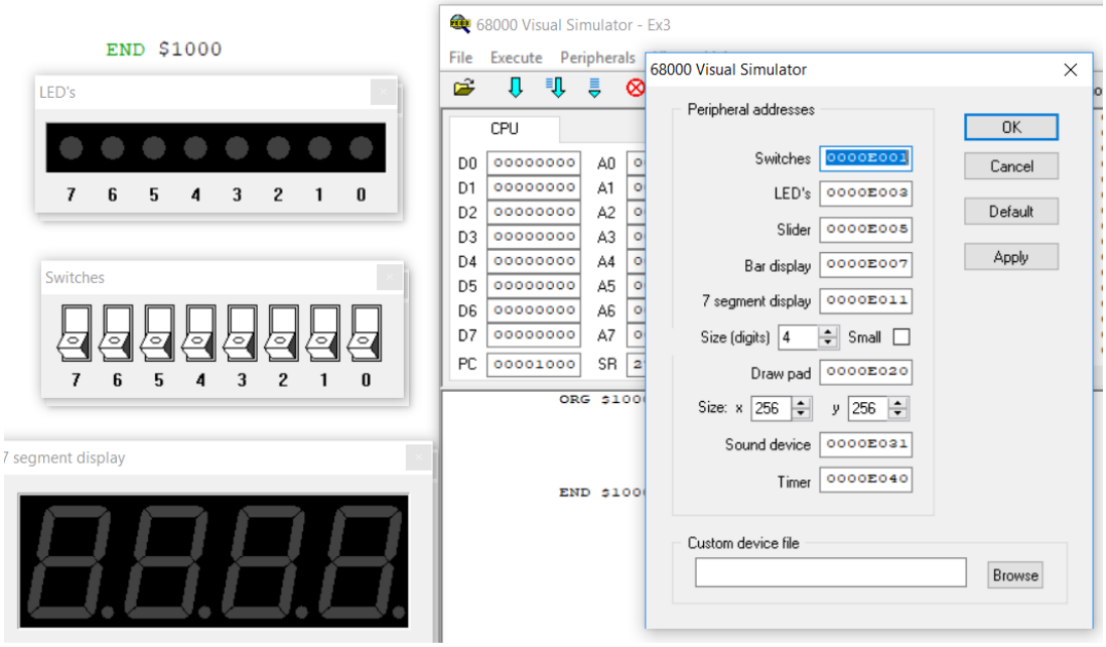
```
ORG $1000
MOVE.L #$12345678,D0
MOVE.W #$8000,A0
MOVE.W #$2000,A0
MOVE.W #$3000,A1
MOVE.B (A0),D0
MOVE.W (A0)+,D0
 MOVE.L (A0)+,D0
MOVE.W D0,-(A1)
MOVE.W $2000,D0
MOVE.L #TABLE,D0
MOVE.W $02(A0),D1
MOVE.L #$02,D2
MOVE.L $02(A0,D2.W),A1
```

```
        ORG $2000
TABLE
        DC.B $11,$22,$33,$44,$55,$66,$77,$88,$99,$AA,$BB,$CC,$DD,$EE
        END $1000
```

Build the code and run it in the simulator showing in each case the addressing mode for source and destination, and the result in affected registers and memories (Draw a table that show the result).

Ex3: Dealing with interfaces::

The simulator has many peripherals (the address of each peripheral is shown in figure, or in help Peripherals Menu).



a) Write a program that read switches and turn on LED0 when switches 0,1 and 7 are on?
b) To write to the first seven-segment we use the address $E011, to the second the address $E013 and so on. Write a program that display the number 15 in the 7 segment display (Note that we must use 7segment code, and not binary code)?
c) Write and run step by step the following code to show its functionality

```
A EQU 9
                ORG $1000
START           MOVE.W #$0,D0
LOOP            ADDI.W #$01,D0
                CMPI.W #A,D0
                BNE LOOP
                BRA START
                END $1000
```

d) Modify your program to show the result in Led (Use step by step and run mode)?
e) Add subroutine delay to your program to correct the effect in run mode?

```
DELAY
        MOVE.L #$000001FF,D2
W2      MOVE.L #$FFFFFFFF,D1
W1      DBRA D1,W1
```

```
DBRA D2,W2
RTS
```

f) Modify your program to display the values in 7 segment display?

Number	g f e d c b a	Hexadecimal
0	0 1 1 1 1 1 1	3F
1	0 0 0 0 1 1 0	06
2	1 0 1 1 0 1 1	5B
3	1 0 0 1 1 1 1	4F
4	1 1 0 0 1 1 0	66
5	1 1 0 1 1 0 1	6D
6	1 1 1 1 1 0 1	7D
7	0 0 0 0 1 1 1	07
8	1 1 1 1 1 1 1	7F
9	1 1 0 1 1 1 1	6F

Ex4: System calls (Trap Functions):

The usual method for calling an operating system in a 68000 system is by means of a special instruction called TRAP. TRAP #15 is used by the simulator for system calls (like BIOS and Int21 interrupts in PC).
To see all system calls you can look to Help > System Calls.
a) Write the following program:

```
ORG $1000
LEA STR,A0
TRAP #15
DC.W 7
MOVE.B #'A',D0
TRAP #15
DC.W 1
TRAP #15
DC.W 0
ORG $2000
STR DC.B 'You Write ',0
END $1000
```

a) Execute the program step by step, and show its functionality (particularly identify which systems calls are used)?
b) Write a program that read a character from keyboard and display it to the screen as following:

```
You Type = X
You Type = Y
….
```

Your program stop when you type $. (Note to return from line to another line you must use PUTCH and print CR (ASCII code 13) and LF (ASCII code 10)).

c) Suppose that PRTSTR (system calls number 7) is not implemented. Program it using PUTCH (system calls number 1)?

d) Write a program that read a number from keyboard and display the following

```
*
**
***
****
.....
```

Ex5: Graphics in 68000IDE simulator:

In 68000IDE (used in lab) we can use Drawpad peripherals to draw graphics. Following table lists the registers used by the drawpad device relative to the base address. The I/O base address is $E020.

offset	function
0	X coordinate
2	Y coordinate
4	command word

Some of the commands are the following (All this commands are in Help> Peripherals Menu see Drawpad):

Erase display

The drawpad is erased when the following command word is written to location **$E024** or a command byte (8 LSB's of the word) to location **$E025**.

0	0	0	0	0	0	0	0	R	G	B	0	0	0	0

ERASE command word

RGB = color to erase drawpad

Erasing the drawpad sets the current pen position to (0,0).

Move to

Under software control, the 68000 first writes X and Y coordinate words to the respective memory locations **$E020** and **$E022** followed by command word $0080 to location **$E024**. (If size is less than 256, coordinate bytes may be written to **$E021** and **$E023**. Equally a command byte $80 written to **$E025**is accepted).

0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

MOVETO command word

Position (0,0) corresponds to the upper-left hand corner of the drawing pad.

Line to

A line is drawn from the previous (X,Y) position (endpoints of MOVETO, LINETO, ELLIPSE or RECTANGLE commands) to the new (X,Y) position, usually the values written to locations **$E020** - **$E022** just before writing the control word. (If size is less than 256, coordinate bytes may be written to **$E021** and **$E023**). Position (0,0) corresponds to the upper-left hand corner of the drawing pad.

The 68000 can draw lines using fifteen different line widths and eight different colors. Color and width are specified in the command word written to **$E024** (or command byte (8 LSB's of word) to **$E025**).

0	0	0	0	0	0	0	0	R	G	B	w	w	w	w

LINETO command word

wwww = line width 1 - 15
RGB = line color

Ellipse

An ellipse (circle) is drawn in the bounding rectangle with the upper left corner from the previous (X,Y) position (endpoints of MOVETO, LINETO, ELLIPSE or RECTANGLE commands) and the lower right corner at the new (X,Y) position, usually the values written to locations **SE020** - **SE022** just before writing the control word. Position (0,0) corresponds to the upper-left hand corner of the drawing pad.

The 68000 can draw ellipses using fifteen different border widths, eight different border colors and eight different filling colors. Colors and width are specified in the command word written to **SE024**.

0	0	0	1	0	r	g	b	0	R	G	B	w	w	w	w

ELLIPSE command word

wwww = border width1 - 15
RGB = border color
rgb = filling color

Rectangle

A rectangle is drawn with the upper left corner from the previous (X,Y) position (endpoints of MOVETO, LINETO, ELLIPSE or RECTANGLE commands) and the lower right corner at the new (X,Y) position, usually the values written to locations **SE020** - **SE022** just before writing the control word. Position (0,0) corresponds to the upper-left hand corner of the drawing pad.

The 68000 can draw rectangle using fifteen different border widths, eight different border colors and eight different filling colors. Colors and width are specified in the command word written to **SE024**.

0	0	1	0	0	r	g	b	0	R	G	B	w	w	w	w

RECTANGLE command word

wwww = border width1 - 15
RGB = border color
rgb = filling color

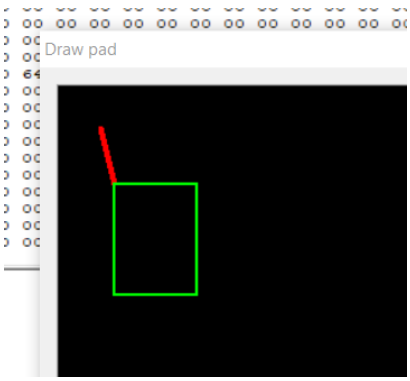
a) Write the following program on simulator:

```
        ORG $1000
DRAWPAD   EQU $E020
X_C       EQU 0
Y_C       EQU 2
COM       EQU 4
        LEA DRAWPAD,A1
        MOVE.W #30,X_C(A1)
        MOVE.W #30,Y_C(A1)
        MOVE.W #$0008,COM(A1)  ; Cursor Move to (30,30)


        MOVE.W #40,X_C(A1)
        MOVE.W #70,Y_C(A1)
        MOVE.W #$0044,COM(A1)  ; Line to (40,70) Color Red   Width 4

        MOVE.W #100,X_C(A1)
        MOVE.W #150,Y_C(A1)
        MOVE.W #$2022,COM(A1)  ; Rectangle to (100,150) Color Red   Width 2
        END $1000
```

Execute the program on simulator to see that it will display:



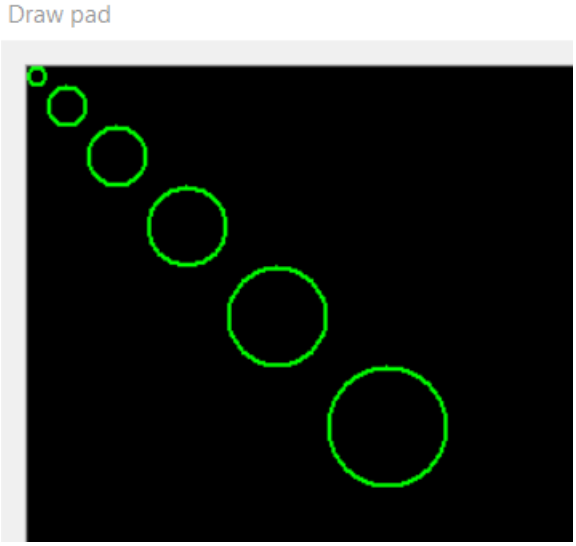
b) Given the following program:

```
        ORG $1000
DRAWPAD EQU $E020
X_C EQU 0
Y_C EQU 2
COM EQU 4
N EQU   70
        LEA DRAWPAD,A1
        MOVE.W #N,D0
        MOVE.W #N,D1
        MOVE.W #N-10,D2

LOOP
        MOVE.W D0,X_C(A1)
        MOVE.W D1,Y_C(A1)
        MOVE.W #$2022,COM(A1)
        ADD.W D2,D0
        ADD.W D2,D1
        SUBI.W #10,D2
        CMPI.W #0,D2
        BNE LOOP
        RTS
        END $1000
```

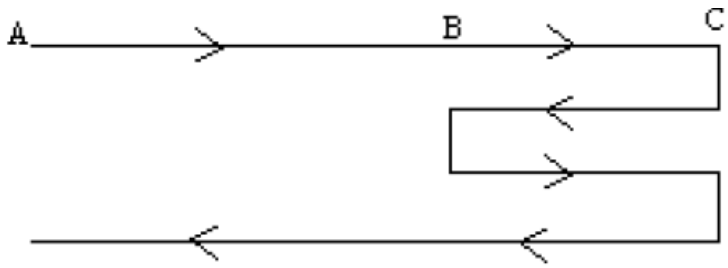
What is the graphics drawn by this program?

c) Modify the program to draw the following:



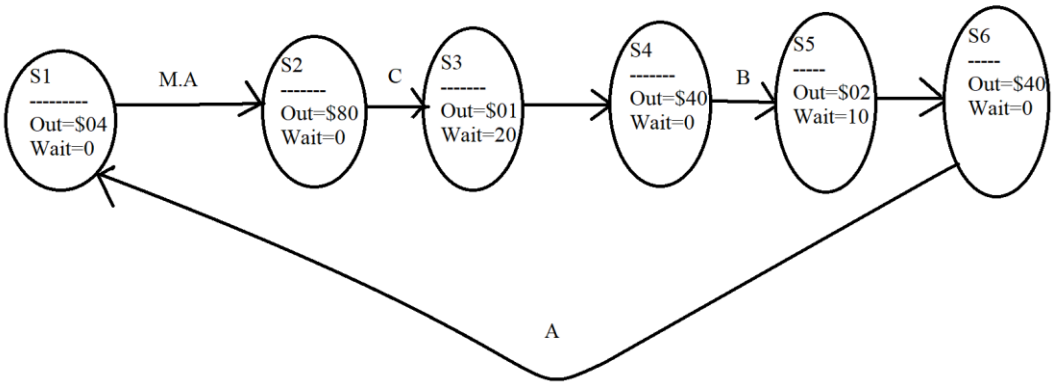
Ex6: Train Command:

Rewrite the train program see in MP1 (Use Switches as inputs and Les as outputs)



Ex7: State Machine:

a) Now we want the train to do the following behavior modeled by a Moore state machine :



As the state machine show the train behavior are the following:

- The train start from station A (state S1)
- When M and A switches are pressed the train start to advance from A to C (modeled by state S2)
- When the train arrive at C it will remain for 20 time (state S3)
- After the time delay it will retreat from C to B (state S4)
- When the train arrive at B it will remain for 10 time (state S5)
- Then the train retreat from B to A (state S6), and when arrived at A the procedure will restart from first state

Note that we will use the Led's as output as following:

- Bit 7 to indicate train Advance
- Bit 6 to indicate train Retreat
- Bit 2 to show that train is in station A
- Bit 1 to show that train is in station B
- Bit 0 to show that train is in station C

Switches are used as inputs as following:

- Bit 3 used to M button
- Bit 2 used station A sensor
- Bit 1 used station A sensor
- Bit 0 used station A sensor

Using the same concept in Slide 157 (FSM implementation) rewrite and test your program?
Use the following code for delay (D0 indicate how many times the delay will be executed):

```
DELAY     ; TIME = D0 SECOND

          CMP.B #0,D0
          BEQ FIN

          MOVE.L #$FFF00000,D2
LOOP
          ADD.L #$01,D2
          BNE LOOP
          ADD.L #$FFFFFFFF,D0
          BNE DELAY
FIN
          RTS
```

b) Modify the program in a) to show graphically (in Drawpad) the train behavior. You must replace sensors in switches by coordinate in the screen as figure indicate.