Activity 2-3 : Central Processing Unit

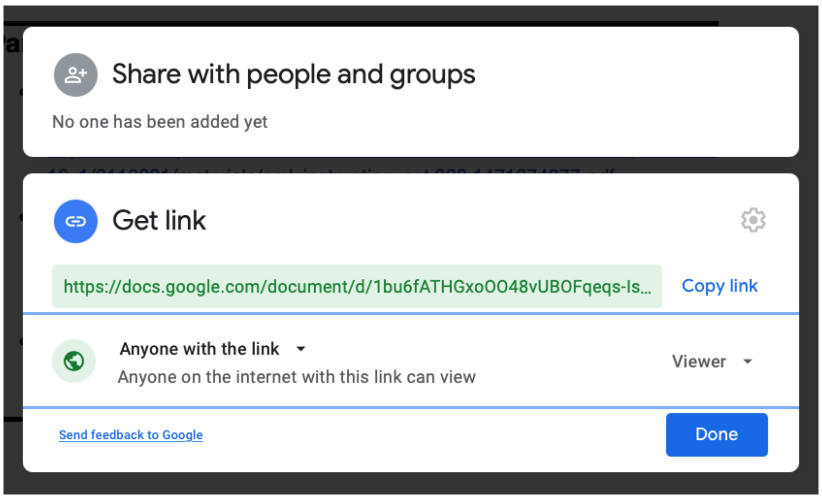
Group No : G-25

Group Member :

1. Phumipat Chaiprasertsud 6432133721
2. Chayut Kritveeranant 6330088721
3. Tanakit Phentun 6330212121
4. Name ID

Part 0 : Preparation

- In part 1, use Activity 2 Reference: SML Instruction Set, which can be downloaded from CourseVille or below: https://www.mycourseville.com/sites/all/modules/courseville/files/uploads/2016_1/2110221/materials/sml_instruction_set.333.1471674877.pdf
- In part 2 and 3, Use Brookshear Simple Machine Emulator to perform the indicated tasks https://www.mycourseville.com/sites/all/modules/courseville/files/uploads/2016_1/2110221/materials/bme.333.1471675276.htm
- Make a copy of this sheet. Answer the questions in the box given. Share this file with the permission for **anyone with link can view the document.** Submit the URL of this file to CourseVille.


Share with people and groups

No one has been added yet

Get link

https://docs.google.com/document/d/1bu6fATHGxoOO48vUBOFqeqs-Is...

Anyone with the link

Anyone on the internet with this link can view

Viewer

Part 3 : Writing A Machine Language Program (40 minutes)

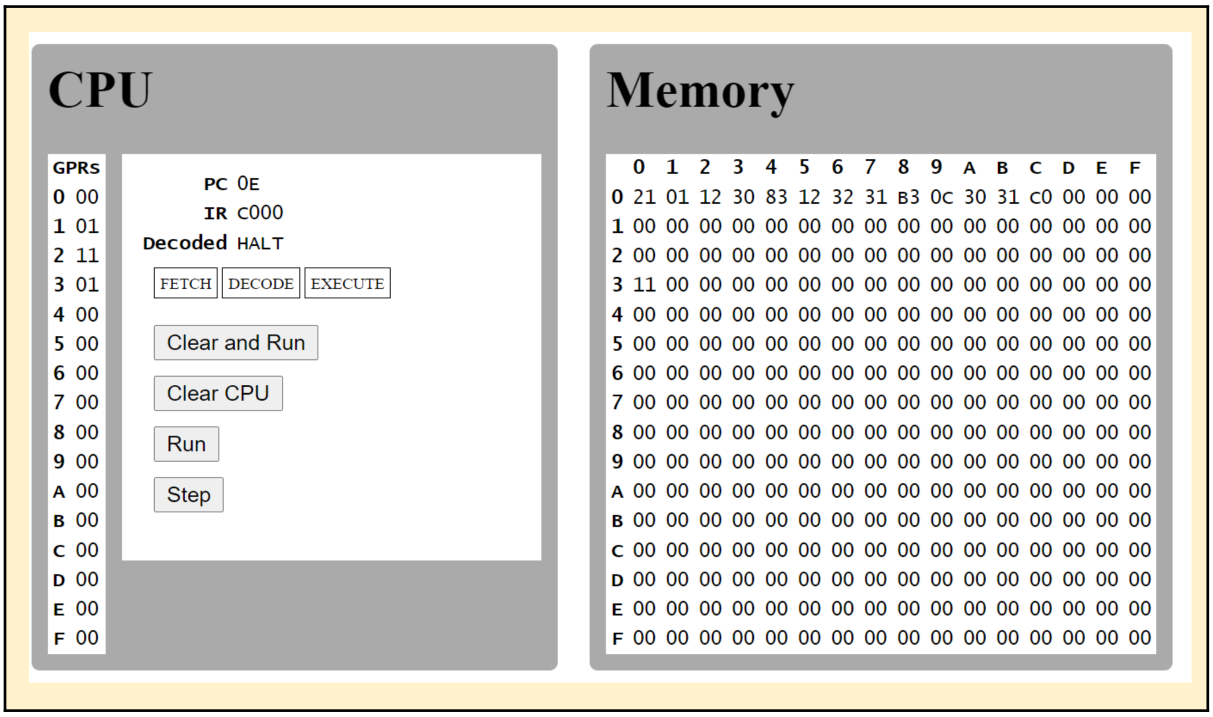
Using the Brookshear Emulator, each group will create 2 programs. To submit the results, students must describe their solutions in a pseudo-code and capture the screenshot of the memory from the emulator that contains the program

1. Write a program in the machine language to check the contents of the memory cell at address 30. If the value is EVEN number, store value of the memory cell at address 30 to the memory cell at address 31. Otherwise, store value 0 to the memory cell at address 31.

pseudo-code of the solution

```
LOADI r1 <- 0x01
LOAD r2 <- *(0x30)
AND r3 <- r1 & r2
STORE r2 -> *(0x31)
JUMP to 0C IF r0 == r3
STORE r0 -> *(0x31)
HALT
```

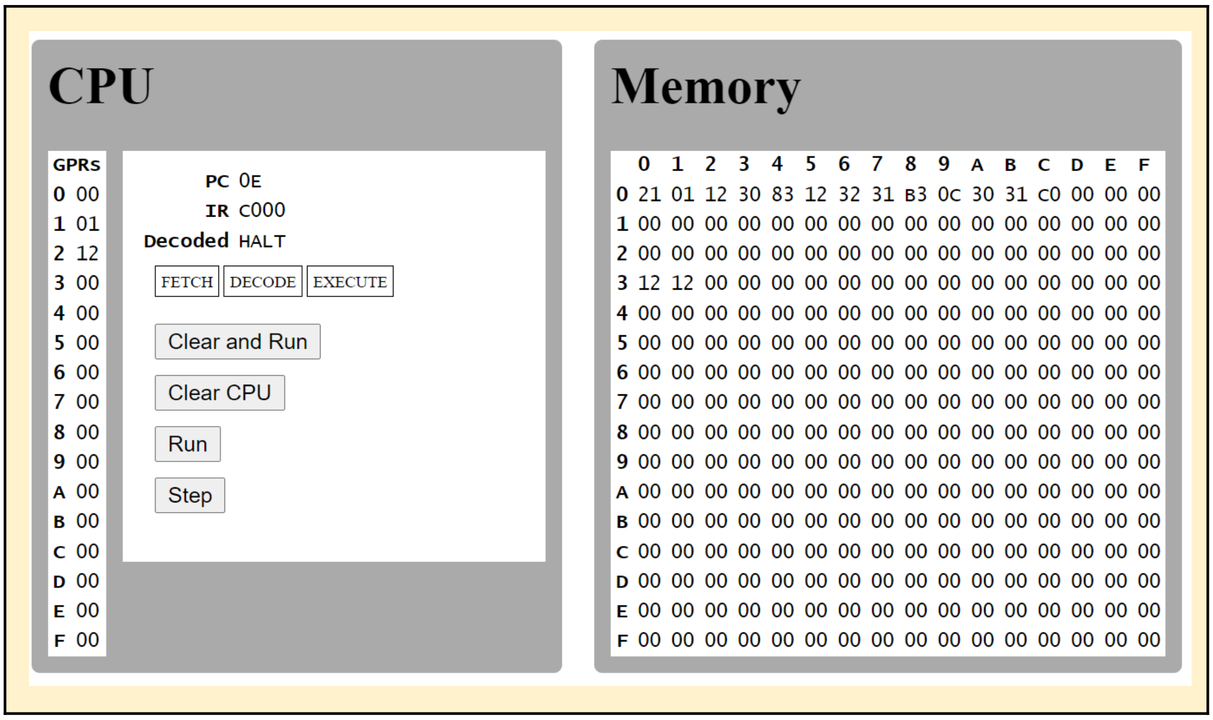
screenshot of memory in the emulator



```
GPRs
0 00
1 01
2 11
3 01
4 00
5 00
6 00
7 00
8 00
9 00
A 00
B 00
C 00
D 00
E 00
F 00

     PC 0E
     IR C000
Decoded HALT
```

```
   0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0 21 01 12 30 83 12 32 31 B3 0C 30 31 C0 00 00 00
1 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
2 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
3 11 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
4 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
5 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
6 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
7 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
9 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
B 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
E 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```



GPRS	
0	00
1	01
2	12
3	00
4	00
5	00
6	00
7	00
8	00
9	00
A	00
B	00
C	00
D	00
E	00
F	00

PC 0E
IR C000
Decoded HALT

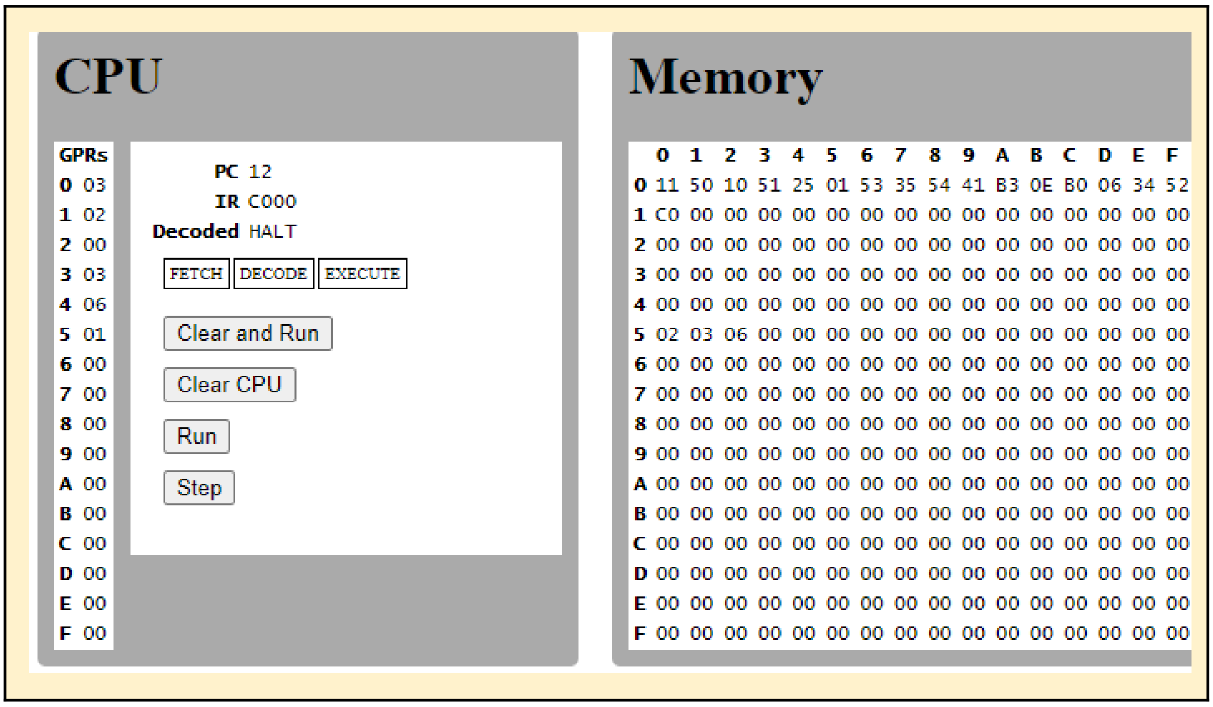
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	21	01	12	30	83	12	32	31	B3	0C	30	31	C0	00	00	00
1	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
2	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
3	12	12	00	00	00	00	00	00	00	00	00	00	00	00	00	00
4	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
5	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
6	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
7	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
9	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

2. Write a program in the machine language to multiply the contents of the memory cell at address 50 to the contents of the memory cell at address 51 and stores the result of multiplication to the memory cell at address 52. You can assume that both values are positive integer and the values are small enough such that the result of multiplication can be stored in 1 byte.

pseudo-code of the solution

```
LOAD r1 <- *(0x50)
LOAD r0 <- *(0x51)
LOADI r5 <- 0x01 // define as 1
ADD r3 <- r3 + r5
ADD r4 <- r4 + r1
JUMP to 0E IF r3 == r0
JUMP to 06
STORE r4 -> *(0x52)
HALT
```

screenshot of memory in the emulator



CPU

GPRs	
0	03
1	02
2	00
3	03
4	06
5	01
6	00
7	00
8	00
9	00
A	00
B	00
C	00
D	00
E	00
F	00

PC 12
IR C000
Decoded HALT

Memory

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	11	50	10	51	25	01	53	35	54	41	B3	0E	B0	06	34	52
1	C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
2	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
3	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
4	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
5	02	03	06	00	00	00	00	00	00	00	00	00	00	00	00	00
6	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
7	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
9	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

A Chance to be “Outstanding” (30 minutes)

Write a program in the machine language to set the contents of the memory cell based on the contents of the memory cell at address F0, F1, and F2 as followed:

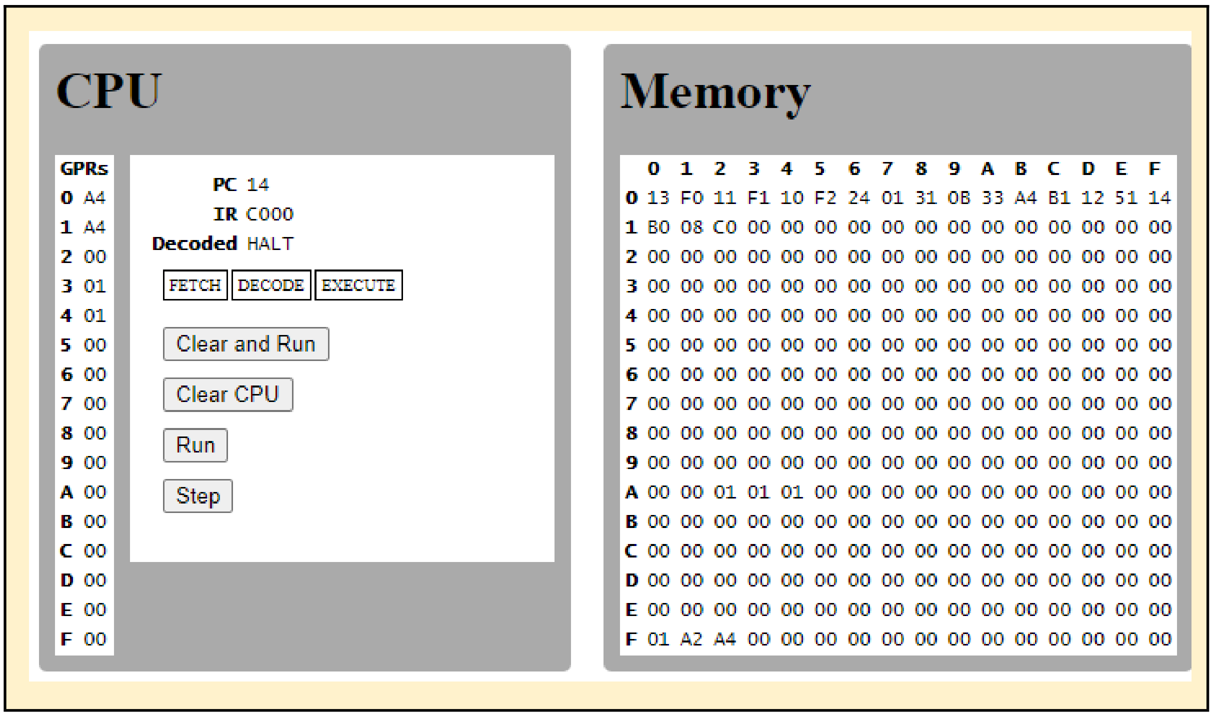
- The content of the memory cell at address F0 defines the value to be filled
- The content of the memory cell at address F1 defines the starting address to be filled
- The content of the memory cell at address F2 defines the last address to be filled

For example, if the contents of the memory cell at address F0, F1, and F2 are 0x01, 0xA2, and 0xA4, respectively. After execution, the program will fill in value 0x01 to memory cell at address A2, A3, and A4

pseudo-code of the solution

```
LOAD r3 <- *(0xF0)
LOAD r1 <- *(0xF1)
LOAD r0 <- *(0xF2)
LOADI r4 <- 0x01 //define as 1
STORE r1 -> *(0x0B)
STORE r3 -> ADDRESS VALUE in r1
JUMP to 12 IF r1 == r0
ADD r1 <- r1 + r4
JUMP to 08
HALT
```

screenshot of memory in the emulator



GPRs	
0	A4
1	A4
2	00
3	01
4	01
5	00
6	00
7	00
8	00
9	00
A	00
B	00
C	00
D	00
E	00
F	00

PC 14
IR C000
Decoded HALT

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	13	F0	11	F1	10	F2	24	01	31	0B	33	A4	B1	12	51	14
1	B0	08	C0	00	00	00	00	00	00	00	00	00	00	00	00	00
2	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
3	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
4	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
5	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
6	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
7	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
9	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
A	00	00	01	01	01	00	00	00	00	00	00	00	00	00	00	00
B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
F	01	A2	A4	00	00	00	00	00	00	00	00	00	00	00	00	00