Machine Language

Coding using Brookshear’s Machine Language

Machine Language

- Computer can only executes programs written in machine language.
- Developing complex programs in machine language is a tedious process.
- Programs are written in high-level language and then translated to the machine language by a program called compiler.

Machine Language

- In this section we will write simple programs in machine language using a simulator program.
- The simulator program is a java program written by Dr.Glenn Brookshear and Dr. Micheal slattery.

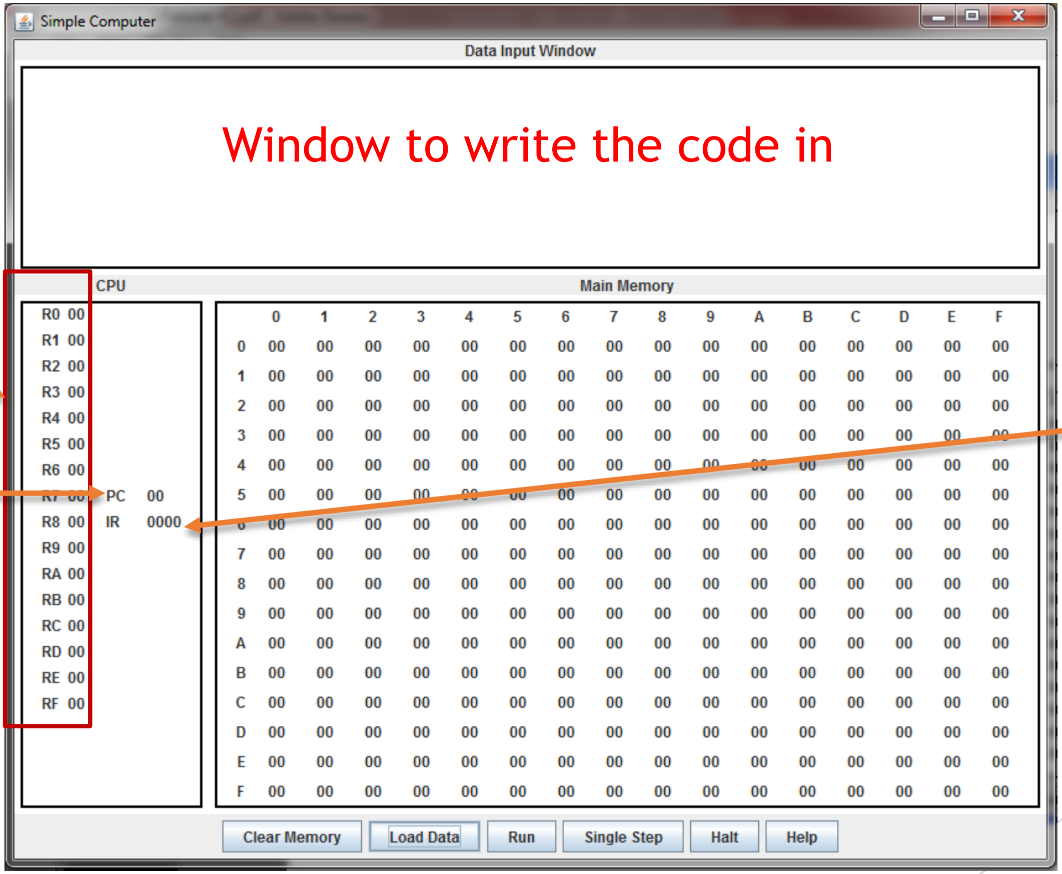
Run the simulator on your PC

Open the command window, then type the following:

- Cd\
- Cd c:/java
- javac Simulator.java
- java Simulator

Run the simulator on your PC

- Now you are able to view the following window



Registers used in the program

Program counter register that contains the memory location where the code line is read from

Instruction register contains the current executed instruction

Op – code	Operand	Description
1	RXY	LOAD register R with the contents of the memory cell at address XY.
2	RXY	LOAD register R with the value XY.
3	RXY	STORE the contents of register R at memory location XY.
4	0RS	MOVE the contents of register R to register S.
5	RST	ADD the integer contents of registers S and T and leave the result in register R. Integers are stored using two's complement notation.
6	RST	ADD the floating-point contents of registers S and T and leave the result in register R.
7	RST	OR the contents of registers S and T and place the result in register R.
8	RST	AND the contents of registers S and T and place the result in register R.
9	RST	EXCLUSIVE OR the contents of registers S and T and leave the result in register R.
A	R0X	ROTATE the contents of register R one bit to the right X times.
B	RXY	JUMP to the instruction located at memory address XY if the contents of register R equals that of register 0.
C	000	HALT

Examples

- The Machine instruction 1234

1 2 3 4

Load Register R with the content of the memory call at address XY

Register R2

Memory location [34]

Load the value stored in Memory location [34] to register R2

Examples

- The Machine instruction 20FF

2 0 FF

Load Register R with the value XY

Register R0

The value = FF

R0=FF

More instructions

- [PC] nn : change the value of register PC to nn so that it starts to read the code from memory location nn
- [Rn] xy: change the value stored in register Rn with the value xy
- [nn] abcde…. : start the line code from the memory Address nn

Example

- Write the following code and run it in the simulation program:
- [00] 20FF   4002   C000
- [PC] 00

- Click on load Data
- Then click run
- What do you expect the output will be:
  - Load R0 with the value FF
  - Move the content of R0 to R2
  - Halt

Experiment1

- Execute the following machine language program.

Address	Content
10	23
11	1F
12	12
13	20
14	50
15	23
16	30
17	40
18	C0
19	00

[10] 231F 1220 5023 3040 C000
[20] 03
[PC]10

Experiment1

- What do you expect the output will be?

[10] 231F 1220 5023 3040 C000

[20] 03

[PC] 10

HALT

R3=1F

R2=[20]

R0=R2+R3

[40]=R0

Experiment1

- Load data and then click single step then record the value of the registers each time: PC,IR,R0,R2,R3,[40],[20]

[10] 231F 1220 5023 3040 C000
[20] 03
[PC]10

PC	IR	R0	R2	R3	[40]	[20]
10	0000	00	00	00	00	03
12	231F	00	00	1F	00	03
14	1220	00	03	1F	00	03
16	5023	22	03	1F	00	03
18	3040	22	03	1F	22	03
1A	C000	22	03	1F	22	03

What is the matlab code for this instructions?

Experiment1

- What is the matlab code for the previous instructions?
  - Y=31+x
  - 31=($1F_{16}$)
  - The value of x is stored in memory location [20]
  - The value of y is stored in the memory location [40]

Experiment1

- Change the memory cell at address 14 to the hexadecimal 90 instead of 50.
- What will be the difference?

[10] 231F 1220 9023 3040 C000
[20] 03
[PC]10

5023 → R0=R2+R3

9023→ R0=R2 xor R3

Address	Content
10	23
11	1F
12	12
13	20
14	~~50~~ 90
15	23
16	30
17	40
18	C0
19	00

Exercise

- Write a program in machine language that finds the 45-36-29. Store the result in memory location 00. Write an explanation of what occurs when the program is run,
- **Hint:**
  - 45= $(2D)_{16}$
  - 36= $(24)_{16}$
  - 29= $(1D)_{16}$

Experiment2

- While the following looks like an illegal algebraic statement, it is a valid Java statement.
- x = x + 5;
- This instruction assigns the sum of 5 and x to x. Write a machine language program for the Simulator, that changes a value stored in memory by adding 5 onto it. Store the program beginning at memory location C2. Assume that x refers to memory location C0. The original value you store in x is up to you.

Experiment2

- Load data and then click single step then record the value of the registers each time: PC,IR,R0,R2,R3,[C0]

[C2] 2305 12CO 5032 30C0 C000
[C0] 03
[PC]C2

PC	IR	R0	R2	R3	[C0]	
C2	0000	0	0	0	03	
C4	2305	0	0	05	03	
C6	12C0	0	03	05	03	
C8	5032	08	03	05	03	
CA	30C0	08	03	05	08	
CC	C000	08	03	05	08	

Experiment2

- Modify the program to accomplish the following:
- x=x+x

5033

```
[C2] 2305 12CO 5032 30C0 C000
[C0] 03
[PC]C2
```

Experiment3 (self study)

- Write a program to find the sum of $7F_{16}$ and 01_{16}, storing the sum at memory location 00. Record your program instructions.

[10] 237F   2201   5023   3000   C000
[PC] 10

Experiment4

Execute the following program code step by step and record the values of R0, R1, R3, PC, IR

[F0] 2000 2101 2305 B3FC 5001 B0F6 C000

[PC] F0

F0	**20**
F1	00
F2	21
F3	01
F4	23
F5	05
F6	B3
F7	FC
F8	50
F9	01
FA	B0
FB	F6
FC	C0
FD	00

Experiment4

Execute the following program code step by step and record the values of R0, R1, R3, PC, IR

[F0] 20 00 21 01 23 05 B3 FC 50 01 B0 F6 C0 00
[PC] F0

	PC	IR	R0	R1	R3
	F0	0000	00	00	00
	F2	2000	00	00	00
	F4	2101	00	01	00
	F6	2305	00	01	05
IF R3=R0 jump to line FC	F8	B3FC	00	01	05
	FA	5001	01	01	05
IF R0=R0 jump to line F6	F6	B0F6	01	01	05
IF R3=R0 jump to line FC	F8	B3FC	01	01	05
	FA	5001	02	01	05
	F6	B0F6	02	01	05
	F8	B3FC	02	01	05
	FA	5001	03	01	05
	. .				
R3=R0 =05 jump to line FC	FC	B3FC	05	01	05
	FE	C000	05	01	05

F0	**20**
F1	00
F2	21
F3	01
F4	23
F5	05
F6	B3
F7	FC
F8	50
F9	01
FA	B0
FB	F6
FC	C0
FD	00

Experiment4

- What changes should be done if it is to be placed in memory starting at location A0

  [F0] 20 00 21 01 23 05 B3 FC 50 01 B0 F6 C0 00
  [PC] F0

  [A0] 20 00 21 01 23 05 B3 AC 50 01 B0 A6 C0 00
  [PC] A0

- Convert it into MATLAB using while loop

F0	**20**	**A0**
F1	00	A1
F2	21	A2
F3	01	A3
F4	23	A4
F5	05	A5
F6	B3	A6
F7	FC	A7
F8	50	A8
F9	01	A9
FA	B0	AA
FB	F6	AB
FC	C0	AC
FD	00	AD

Experiment4

- Convert it into MATLAB using while loop:

```
Count=0;
While (count ~=5 )
        count=count+1;
end
```

Exercise

- Convert the following Matlab code into machine instruction:
  - Assume x is stored in memory location 1D and y is stored in memory location 1C, x=1 and y=5.

```
While (y>=x)
        x=x+1
end
```

Solution

- [00] 111D 121C 23FF 2401 9513 5554 2080 8550 B516 5114 B008 311D C000
- [PC] 00
- [1D] 01
- [1C] 05

Exercise

- There is no MULTIPLY instruction in Brookshear's Machine Language. Using the available operations, write a program that multiplies 4 x 6 and stores the product in memory location 00. Hint: **A loop is needed**. Write an explanation of what occurs when the program is run,

Solution

- [00] 2004 2100 2200 2306 2306 2401 B112 5223 5114 B00A C000

Non Circular Left shift

- Logic Shift
  - Shift left: push the pattern to the left and ADD 0 to the least significant bit

~~1~~0110010

Left shift

- Multiplication with 2^n where n is the number of bits to shift.
- Ex1: 3 X 8 = 3 X 2^3 = 24
  - (3 bits shift)
  - 0000 0011
  - 0000 0110
  - 0000 1100
  - 0001 1000 → 16+8=24

Left shift (self Read)

- Multiplication with 2^n where n is the number of bits to shift.
- Ex2: 5 X 16 = 5 X 2^4 = 80
  - (4 bits shift)
  - 0000 0101
  - 0000 1010
  - 0001 0100
  - 0010 1000
  - 0101 0000 ➔64+16=80

Non Circular right shift

- Logic Shift
  - Shift right: push the pattern to the right and add zero to the most significant bit

0101100~~1~~

Right shift

- Division with 2^n where n is the number of bits to shift.
- Ex1: $16/4 = 16 \ / \ 2^2 = 4$
  - (2 bits shift)
  - 0001 0000
  - 0000 1000
  - 0000 0100 → 4

Non Circular Shift (self Read)

- Arithmetic Shift: Arithmetic shift is used in signed numbers to maintain the sign
  - Shift right

11011001

Rotation (Right Rotation)

- Circular shift
- Example: 1011001
- Right Rotate 1 bit

1011001 → 1101100

Rotation (left Rotate)

- Circular shift
- Example: 1011001
- left Rotate 1 bit

1011001 → 0110011

Op – code	Operand	Description
1	RXY	LOAD register R with the contents of the memory cell at address XY.
2	RXY	LOAD register R with the value XY.
3	RXY	STORE the contents of register R at memory location XY.
4	0RS	MOVE the contents of register R to register S.
5	RST	ADD the integer contents of registers S and T and leave the result in register R. Integers are stored using two's complement notation.
6	RST	ADD the floating-point contents of registers S and T and leave the result in register R.
7	RST	OR the contents of registers S and T and place the result in register R.
8	RST	AND the contents of registers S and T and place the result in register R.
9	RST	EXCLUSIVE OR the contents of registers S and T and leave the result in register R.
A	R0X	ROTATE the contents of register R one bit to the right X times.
B	RXY	JUMP to the instruction located at memory address XY if the contents of register R equals that of register 0.
C	000	HALT

Example (self Read)

- Right rotate the following bits 3 times

1011001

1101100

0110110

0011011

Example (self Read)

- Right Rotate the following pattern 3 times:
- 1110 1010
- 0111 0101
- 1011 1010
- 0101 1101

Left Rotate (self Read)

- Left Rotate the following pattern 3 times:
- 1110 1010
- 1101 0101
- 1010 1011
- 0101 0111

- How to solve it using Right Rotation????

Left Rotate using Brookshear's Simulator

- Left rotate the following bits 3 times = Right rotate 5 times
- (number of bits – number of left rotations) = 8-3=5
  - 1110 1010
  - 0111 0101
  - 1011 1010
  - 0101 1101
  - 1010 1110
  - 0101 0111
- They are Similar

Right shift (self Read)

- Right shift the following pattern 3 times
- 1110 1010
- 0111 0101
- 0011 1010
- 0001 1101
- How to solve it using Right Rotation????

Right shift (self Read)

- Rotate Right 3 times THEN AND with 1F ‘ put zeros in the bits we want to remove’
- 1110   1010
- 0111   0101
- 1011   1010
- 0101   1101

- 0101   1101
- 0001   1111 (AND)
- __________
- 0001   1101

Left shift (self Read)

- Left shift the following pattern 3 times
- 1110 1010
- 1101 0100
- 1010 1000
- 0101 0000

- How to solve it using Right Rotation????

Left shift (self Read)

- Rotate Right (8-3=5) times THEN AND with F8 (1111 1000)
  - 1110 1010
  - 0111 0101
  - 1011 1010
  - 0101 1101
  - 1010 1110
  - 0101 0111

- 0101 0111
- 1111 1000(AND)
- ___________
- 0101 0000

Exercise

- Now solve this operation without using loop:
- **write a program that multiplies 6 x 4 and stores the product in memory location 00.**

Solution

- Now solve this operation without using loop:
- **write a program that multiplies 6 x 4 and stores the product in memory location 00.**
- **Hint:** 6X 2^2 (shift left 2 times)
- Rotate Right 8-2=6 times THEN AND with FC 'we want the last 2 bits 0'

[A0] 2106 A106 22FC 8312 3300
[PC]A0

Exercise

- Write a program in machine instructions that would divide 30 by 4 (in decimal notation). Using **shift operation**

Masking Operation

Exercise

- Using the machine language, write programs to perform each of the following tasks:
  - Change the least significant 4 bits in the memory cell at location 34 to 0s while leaving the other bits unchanged.
  - Copy the least significant 4 bits from memory location A5 into the least significant 4 bits of location A6 while leaving the other bits at location A6 unchanged.
  - Put 1s in all but the most significant bit of an 8-bit pattern without disturbing the most significant bit.
  - Complement the most significant bit of an 8-bit pattern without changing the other bits.
  - Copy the bit pattern stored in memory location 44 into memory location AA.

Exercise

- Describe a sequence of logical operations (along with their corresponding masks) that, when applied to an input string of 8 bits, produces an output byte of all 0s if the input string both begins and ends with 1s.

Solution

- [00] 2081 8320 B308 B00C 2400 8224 C000