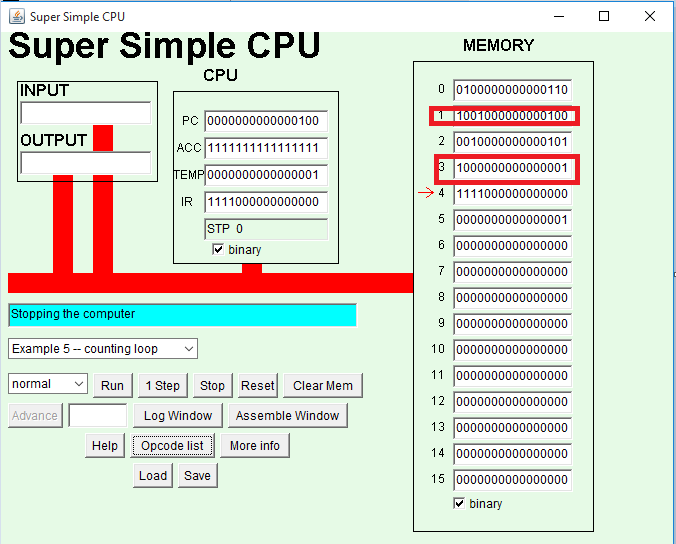
**LAB 5**

1. Lab 7 Computer Cycling (Lab7\_Manual.pdf)

a. Exercise 4

1. Start the app.
2. Load the counting loop example.
3. Run the app and write down the PC after each instruction.
4. Take a screenshot and circle the all jumping instructions.



1. As a human “disassembler”. Translate the comp instructions into in cell1 to 5 in mnemonics.

Mem Cell Mnemonic

0 LDI

1 JNG

2 SUB

3 JMP

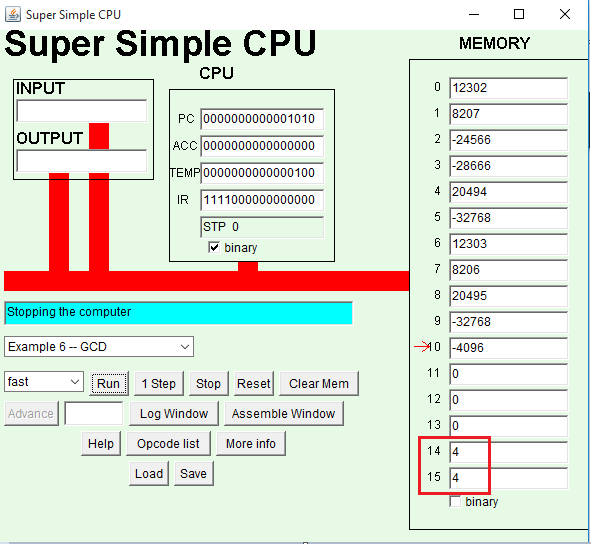
4 STP

b. Exercise 5

1. Start the app.
2. Load the GCD example.
3. Run it.
4. Type 16 in location 14 and 28 in location 15. What would be the GCD of these numbers?

ANS: GCD of two numbers is 4.

1. Run the program and take a screenshot.



1. What is the GCD of 30 and 31? Type 30 in cell 14 and 31 in cell 15?

ANS: GCD of 30 and 31 is 1.

c. Exercise 6

1. Start the app.
2. Load the GCD example.
3. Disassemble the program in memory into human readable code by decoding the instructions.

Ans:

0 LOD A

1 SUB B

2 JZR 10

3 JNG 6

4 STO A

5 JMP 0

6 LOD A

7 SUB B

8 STO A

9 JMP 0

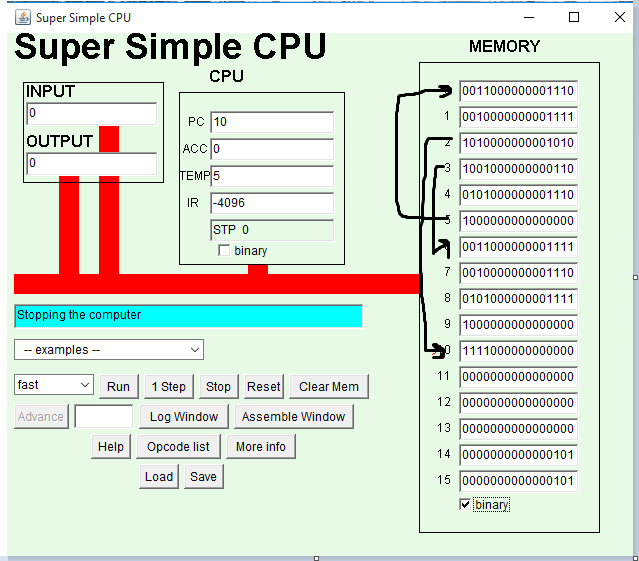
10 STP

2. Lab 11 Low-Level Languages (Lab11\_Manual.pdf)

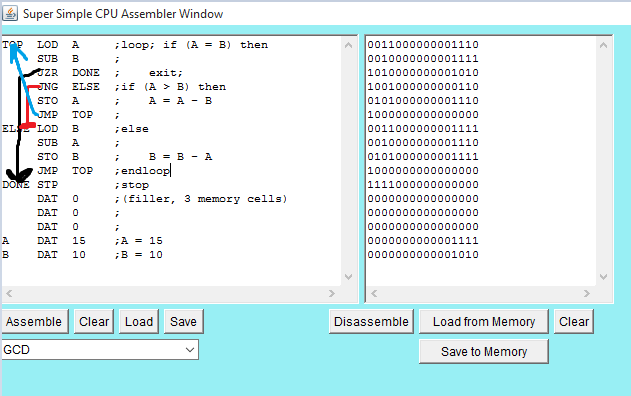
a. Exercise 1

1. Start the app.
2. Open the assembler Window.
3. Load the GCD example and take a screenshot.
4. On your screenshot, draw arrows from the jumping instructions to target address.

ANS:



In the assemble window it is like:



1. Change the second line from the bottom to read A DAT 15; A=15. Change the bottom line B DAT 10; B= 10.
2. Click Assemble, then click Save to Memory. Go back to the main window.
3. Trace the program. In other words pretend you are a computer and do one instruction at a time. Writing down the values in IR and ACC at the end of each fetch execute cycle.

First, note the number in PC, click on the 1 step button. When it finishes running, note the decoded IR and the value in the accumulator. Start a new line for the next step, noting the PC first, then once again clicking the 1 step button and noting the resulting IR and ACC values. Continue until the program is finished.

|  |  |  |
| --- | --- | --- |
| PC | IR | ACC |
| 0 | LOD 14 | 15 |
| 1 | SUB 15 | 5 |
| 2 | JZR 10 | 5 |
| 3 | JNG 6 | 5 |
| 4 | STO 14 | 5 |
| 5 | JMP 0 | 5 |
| 0 | LOD 14 | 5 |
| 1 | SUB 15 | -5 |
| 2 | JZR 10 | -5 |
| 3 | JNG 6 | -5 |
| 6 | LOD 15 | 10 |
| 7 | SUB 14 | 5 |
| 8 | STO 15 | 5 |
| 9 | JMP 0 | 5 |
| 0 | LOD 14 | 5 |
| 1 | SUB 15 | 0 |
| 2 | JZR 10 | 0 |
| 10 | STP 0 | 0 |

1. Now click on the Log window and compare your trace with what the computer did as it ran the program. Did you get the same results?

ANS: Yes, the results that I traced are completely same as shown in the log window.

b. Exercise 3

1. Start the app.
2. Click on assembler window button.
3. In the text area, type the program.
4. Click on the assembler button in the blue window. What do you see in the right text area. Can you make out the relationship between the mnemonics in the left text area and the binary codes in the right?

ANS: In the right text area, the following binary code is shown:

0110000000000000

0101000000001010

0010000000001011

1010000000000111

0100000000000000

0111000000000000

1111000000000000

0100000000000001

0111000000000000

1111000000000000

0000000000000000

0000000000000101

0000000000000000

0000000000000000

0000000000000000

0000000000000000

There is a very clear relationship between the mnemonics in the left text area and the binary codes in the right that the first four digits of each binary code represents the mnemonic instructions at the corresponding lines of the left text area. The line number 11 and 12 shows the data that means it stores the value of A and FIVE.

1. To transfer this program into memory of the super simple CPU, Click on the button and save to the Memory. Close the blue window. Notice that the binary version of the program has been copied into the memory cells.
2. Run the program. The first instruction will ask you for input. Type in “101”, since Super Simple CPU requires your input to be binary. What is the output?

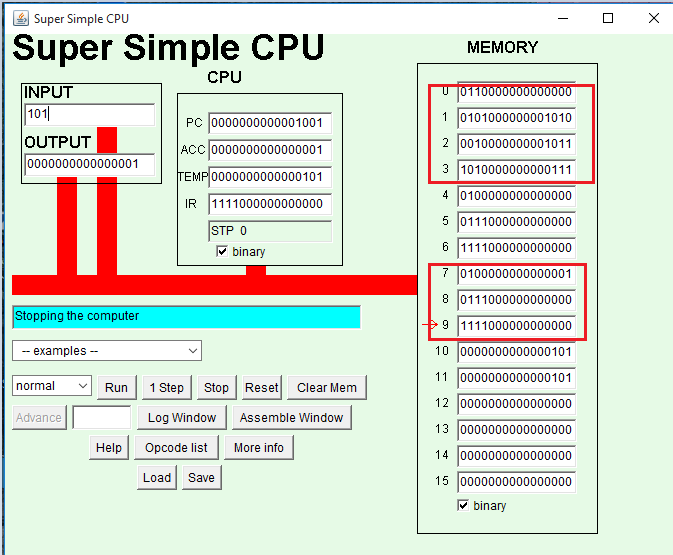
ANS: Output is 0000000000000001.

1. Click on reset and run the program again. This time, type in “111” and write down what the output is.

ANS: Output is 0000000000000000.

1. Look at the previous program and trace through the instructions. Imagine you typed “101”. Circle all the instructions that are executed when this is what you inputted.

Ans: The instructions enclosed in the red squares are executed:



1. Two fundamental control structures in computer programming are decisions and loops. Does this program contain a decision? If so, which instructions trigger the taking of an alternative path?

ANS: This program contains decisions at the instructions “JZR SHOW1” which means jump to SHOW1 if the value in accumulator is less than 0. At this place, when the value in accumulator is less than 0 then it jumps to the SHOW1 instruction.

1. Does this program contain a loop? If so, where is the instruction that causes the computer to jump back to an earlier spot?

ANS:

3.(Counting Loop) Write a pseudocode that performs example 5 of Super Simple CPU Applet.

ANS: We have a value in memory cell 5 so let us call it X

|  |  |  |  |
| --- | --- | --- | --- |
| Assembly | ACC | X | Pseudocode |
| LDI 6 | 6 | 1 | Set A to 6 |
| JNG 4 | 5 | 1 | IF (A<0) |
| SUB X | 6-1 = 5 | 1 | Set A to A - X |
| JMP | 5 | 1 | ELSE |
| SUB X | 5-1=4 | 1 | Set A to A - X |
| JMP | 4 | 1 |  |
| SUB X | 4-1= 3 | 1 | Set A to A – X |
| JMP | 3 | 1 |  |
| SUB X | 3-1 | 1 | Set A to A - X |
| JMP | 2 | 1 |  |
| SUB X | 2-1 | 1 | Set A to A - X |
| JMP | 1 | 1 |  |
| SUB X | 1-1 | 1 | Set A to A -X |
| JMP | 0 | 1 |  |
| SUBX | 0-1 | 1 | Set A to A - X |
| JMP | -1 | 1 |  |
| JNG | -1 | 1 | IF (A<0) |
| STP | -1 | 1 | - |

Pseudocode is:

Set X to 1

Set A TO 6

IF (A>0)

Set A to A - X

END IF

4. Given the following "Super Simple CPU" program, consider three different values for INPUT (1, 2, or 3). For each input value, trace the code by writing the values of Accumulator, Memory (ONE), Memory (TWO), and OUTPUT in a table and recording their changes after each instruction. You should have 3 trace tables for 3 different values of input.

INPUT ; consider three different values for input (1,2,3)

SUB ONE

JZR ZERO

SUB TWO

JNG NEG

ADD TWO

OUT

STP

NEG ADD TWO

OUT

STP

ZERO OUT

STP

ONE DAT 1 ; A data value, the constant 1

TWO DAT 2 ; A data value, the constant 2

ANS: For input =1, output = 0

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Label | Instruction | Input | Accumulator | ONE | TWO | OUT |
|  | INPUT | 1 | 1 | 1 | 2 |  | |
|  | SUB ONE |  | 1-1=0 | 1 | 2 |  | |
|  | JZR ZERO |  | 0 | 1 | 2 |  | |
|  | SUB TWO |  |  |  |  |  | |
|  | JNG NEG |  |  |  |  |  | |
|  | ADD TWO |  |  |  |  |  | |
|  | OUT |  |  |  |  |  | |
|  | STP |  |  |  |  |  | |
| NEG | ADD TWO |  |  |  |  |  | |
|  | OUT |  |  |  |  |  | |
|  | STP |  |  |  |  |  | |
| ZERO | OUT |  | 0 | 1 | 2 | 0 | |
|  | STP |  | 0 | 1 | 2 | 0 | |
| ONE | DAT 1 |  |  |  |  |  | |
| TWO | DAT 2 |  |  |  |  |  | |

When input is 2, output is 1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Label | INSTRUCTIONS | Input | ACCUMULATOR | ONE | TWO | OUT |
|  | INPUT | 2 | 2 | 1 | 2 |  |
|  | SUB ONE |  | 2-1=1 | 1 | 2 |  |
|  | JZR ZERO |  |  |  |  |  |
|  | SUB TWO |  | 1-2=-1 | 1 | 2 |  |
|  | JNG NEG |  |  |  |  |  |
|  | ADD TWO |  |  |  |  |  |
|  | OUT |  |  |  |  |  |
|  | STP |  |  |  |  |  |
| NEG | ADD TWO |  | -1 + 2=1 | 1 | 2 |  |
|  | OUT |  | 1 | 1 | 2 | 1 |
|  | STP |  | 1 | 1 | 2 | 1 |
| ZERO | OUT |  |  |  |  |  |

When input = 3, output is 2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| LABEL | INSTRUCTIONS | INPUT | ACCUMULATOR | ONE | TWO | OUT |
|  | INPUT | 3 | 3 | 1 | 2 |  |
|  | SUB ONE |  | 3-1 = 2 | 1 | 2 |  |
|  | JZR ZERO |  |  |  |  |  |
|  | SUB TWO |  | 2-2=0 | 1 | 2 |  |
|  | JNG NEG |  |  |  |  |  |
|  | ADD TWO |  | 0+2=2 | 1 | 2 |  |
|  | OUT |  | 2 | 1 | 2 | 2 |
|  | STP |  | 2 | 1 | 2 | 2 |
| NEG | ADD TWO |  |  |  |  |  |
|  | OUT |  |  |  |  |  |
|  | STP |  |  |  |  |  |
| ZERO | OUT |  |  |  |  |  |
|  | STP |  |  |  |  |  |

5. Given the following "Super Simple CPU" program:

a. Trace the code by writing the values of Accumulator, Memory (ONE), and OUTPUT in a table (shown below) and recording their changes after each instruction.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Label | Instructions | Accumulator | ONE | OUT |
|  | LDI 6 | 6 | 1 | - |
| COUNT | JZR DONE | - | - | - |
|  | OUT | 6 | 1 | 6 |
|  | SUB ONE | 6-1=5 | 1 | - |
|  | JMP COUNT |  |  |  |
| DONE | STP | - | - | - |
|  | DAT 1 |  |  |  |

This program will continue until accumulator is 0.

b. Explain the function of the program in one sentence (Hint: For the function of the program, you can explain the output of the program).

ANS:

LDI 6

COUNT JZR DONE

OUT

SUB ONE

JMP COUNT

DONE STP

ONE DAT 1

ANS: The program loads the accumulator and subtracts 1 from it until the value in accumulator becomes 0.

Set A to 6

IF (A!=0)

Set OUT to A

Set A to A-ONE

END IF

6. Given the following pseudocode:

a. Trace the program by writing the values of variables, conditions, and printout in a table (shown below).

b. Transform the pseudocode to a program in Super Simple CPU Assembly language.

c. Explain the function of the program in one sentence. (Hint: For the function of the program, you can explain the printout of the program).

Set num to 10

WHILE (num is greater than 0)

Print num

Set num to num – 2

END WHILE

ANS:

a.

|  |  |  |
| --- | --- | --- |
| num | num>0 | printout |
| 10 | true | 10 |
| 8 | true | 8 |
| 6 | true | 6 |
| 4 | true | 4 |
| 2 | true | 2 |
| 0 | false | - |

b. The program is as follows:

TOP LOD num

JZR DONE

JNG DONE

OUTPUT

SUB X

JMP TOP

DONE STP

num DAT 10

X DAT 2

c. It prints the value of num until it becomes zero or from the printout it can be said that it prints the even numbers from 2 to 10 in descending order.

7. Given the following pseudocode, assume the sequence of input numbers is: 4, 5, -1, 2, -4, -2, 6, 8, -1, 0. (Note: The program may not read all these numbers).

a. Trace the program by writing the values of variables, conditions, and printout in a table (shown below).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| sum | count | Count<=4 | number | Number>0 | printout |
| 0 | 1 | true | 4 | true | - |
| 4 | 2 | true | 5 | true | - |
| 9 | 3 | true | -1 | false | - |
| 9 | 4 | true | 2 | true | - |
| 11 | 5 | false | - | - | Sum is 11 |

b. How many numbers does the program read?

ANS: The program reads the four numbers.

c. Explain the function of the program in one sentence. (Hint: For the function of the program, you can explain the printout of the program in terms of input values).

ANS: This program will read the first four numbers of the given input and add the all positive values and print it as sum.

The program is as follows:

Set sum to 0

Set count to 1

WHILE (count <= 4)

Read number

IF (number > 0)

Set sum to sum + number

Increment count

END IF

END WHILE

Write "Sum is " + sum