

ELEMENTS OF COMPUTING SYSTEM-

2

PROJECT WORK

Team Members:

- 1)Garikipati Karthik – AIE22020
- 2)Vinitha Chowdary – AIE22066
- 3)Chakravaram HariPriya -AIE22008

Aim: Write a code to convert VM code to Assembly code

Basic: Assembly code

The screenshot displays a virtual machine emulator interface. At the top, there is a toolbar with icons for file operations, navigation, and execution controls (Slow, Fast). Below the toolbar, the interface is divided into several panels:

- ROM Panel:** A list of memory addresses from 585 to 613. Address 600 is highlighted in yellow.
- RAM Panel:** A table showing memory addresses and their values. Address 0 contains the value 257, which is highlighted in yellow.
- Code Panel:** A window displaying assembly code. The code includes comments for setting up stack pointers and base addresses, a loop labeled 'repeat 600', and an 'output;' statement. The 'output;' line is highlighted in yellow.
- ALU Panel:** A diagram of the Arithmetic Logic Unit. It shows a green trapezoidal block labeled 'M+1'. The 'D Input' is 0 and the 'M/A Input' is 256. The 'ALU output' is 257.

At the bottom, there are input fields for 'PC' (Program Counter) with the value 600 and 'A' (Accumulator) with the value 0.

POINTER: ASSEMBLY OUTPUT

The screenshot displays a computer architecture simulator interface. At the top, a toolbar includes icons for file operations, navigation, and execution controls. Below the toolbar, the main workspace is divided into several panels:

- ROM Panel:** A list of memory addresses from 435 to 463. Address 450 is highlighted in yellow.
- RAM Panel:** A list of memory addresses from 0 to 28. Address 0 is highlighted in yellow and contains the value 257.
- Assembly Code Panel:** Displays the following code:

```
load PointerTest.asm,  
output-file PointerTest.out,  
compare-to PointerTest.cmp,  
output-list RAM[256]%D1.6.1 RAM[3]%D1.6.1  
          RAM[4]%D1.6.1 RAM[3032]%D1.6.1 RAM[3046]%D1.6.1;  
  
set RAM[0] 256, // initializes the stack pointer  
  
repeat 450 { // enough cycles to complete the execution  
    ticktock;  
}  
  
// outputs the stack base, this, that, and  
// some values from the the this and that segments  
output;
```

The line `output;` is highlighted in yellow.
- PC (Program Counter):** A register showing the value 450.
- A (Accumulator):** A register showing the value 0.
- D (Data Register):** A register showing the value 46.
- ALU (Arithmetic Logic Unit):** A component with a green triangle labeled 'M+1'. It has two inputs: 'D Input' (46) and 'M/A Input' (256). The 'ALU output' is 257.

Static: Assembly output

The screenshot displays a static assembly simulator interface. At the top is a toolbar with icons for file operations, navigation, and execution controls (Slow/Fast). Below the toolbar are three main panels: ROM, RAM, and a code editor.

ROM Panel: A list of memory addresses from 180 to 208. Address 200 is highlighted in yellow.

RAM Panel: A table of memory addresses and their values.

Address	Value
0	257
1	300
2	400
3	3030
4	3040
5	0
6	510
7	0
8	0
9	0
10	0
11	0
12	0
13	0
14	0
15	0
16	888
17	333
18	111
19	0
20	0
21	0
22	0
23	0
24	0
25	0
26	0
27	0
28	0

Code Editor: Contains assembly code for a program named StaticTest.tst.// by Nisan and Schocken, MIT Press.
// File name: projects/07/MemoryAccess/StaticTest/StaticTest.tst

load StaticTest.asm,
output-file StaticTest.out,
compare-to StaticTest.cmp,
output-list RAM[256]%D1.6.1;

set RAM[0] 256, // initializes the stack pointer

repeat 200 { // enough cycles to complete the execution
 ticktock;
}

output; // the stack base

Registers and ALU: The PC register is at 200, and the A register is at 0. The ALU shows a D input of 888 and an M/A input of 256, resulting in an output of 257.

Simple add: Assembly output

The screenshot displays a computer architecture simulator interface. At the top, a toolbar includes icons for file operations and execution controls (Slow, Fast, Program flow, Script, Decimal). Below the toolbar, the interface is divided into several sections:

- ROM:** A table of memory addresses and their corresponding assembly instructions. A "Load Program" button is visible next to it.
- RAM:** A table of memory addresses and their current values.
- Assembly Code:** A text area showing the assembly code being executed, including comments and instructions like `load SimpleAdd.asm`, `output-file SimpleAdd.out`, `compare-to SimpleAdd.cmp`, `output-list RAM[0]&D2.6.2 RAM[256]&D2.6.2`, `set RAM[0] 256`, `repeat 60`, `ticktock`, and `output`.
- Registers:** A section showing the current values of the Program Counter (PC) and the Accumulator (A).
- ALU:** A section showing the ALU input (D Input: 8, M/A Input: 256) and the ALU output (257).

The ALU output is 257, which is the sum of the D Input (8) and the M/A Input (256).

Stack: Assembly Output

The screenshot displays a computer architecture simulator interface. At the top, a toolbar includes icons for file operations, navigation (single and double arrows), a stop button, and a speaker icon. It also features a speed slider between 'Slow' and 'Fast', and three dropdown menus: 'Animate: Program flow', 'View: Script', and 'Format: Decimal'.

On the left, the 'ROM' panel shows a list of memory addresses from 1276 to 1304. Address 1304 is highlighted in yellow. Below this panel, the 'PC' (Program Counter) is shown with the value 1304.

In the center, the 'RAM' panel shows a list of memory addresses from 0 to 28. Address 0 is highlighted in yellow and contains the value 267. Below this panel, the 'A' (Accumulator) register is shown with the value 0.

On the right, a code window displays assembly code. The code includes comments and instructions for setting up a stack, repeating a loop 1000 times, and outputting stack contents. The instruction `output-list RAM[261]%D2.6.2 RAM[262]%D2.6.2 RAM[263]%D2.6.2 RAM[264]%D2.6.2` is highlighted in yellow. Below the code window, a 'D' (Data) register is shown with the value 90.

At the bottom right, the 'ALU' (Arithmetic Logic Unit) is shown. It has two inputs: 'D Input' with the value 90 and 'M/A Input' with the value 266. The ALU contains a green trapezoidal block labeled 'M+1'. The 'ALU output' is shown as 267.