Sogang University
CSE3030 - Introduction to Computer Systems

Introduction to Computer Systems
Ex. – Homework2: MU0 Assembler & Simulator

Making Assembler of 16bit Processor

1. Introduction

Assembler is one of the phases in the compilation system. Assembler creates object codes from assembly program and there exist various assemblers for each processor like x86, ARM, etc. In this project, students are required to implement an assembler and simulator for the MU0 processor. Because MU0 is a simple processor with a small instruction set, an assembler and a simulator can be easily implemented. We hope that students become familiar with the C language and learn the compilation steps from assembly language to object codes and MU0 instructions via implementing and testing this project.

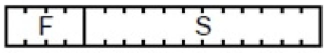
2. Background

This project requires students to implement a compiler and a simulator for mu0. Before implementing those programs, students need to know about mu0. Although mu0 was already handled in a class, this documentation describes mu0 briefly and only necessary parts, especially the mu0 instruction set. And then 2-pass assembler will be explained because students should implement the mu0 compiler as a 2-pass assembler.

a) MU0

MU0 is a simple processor. It has a simple architecture and instruction set. MU0 is a 16-bit machine. The address space of MU0 is 12 bits long so 4K words can be handled. And it has only two registers: PC (Program Counter) and ACC (Accumulator, the only available user register)

Instruction Format



- F: 4-bit instruction
- S: 12-bit operand address

Instruction Set

F	Mnemonic	Description
0	LDA S	ACC := [S]
1	STO S	[S] := ACC
2	ADD S	ACC := ACC + [S]
3	SUB S	ACC := ACC - [S]
4	JMP S	PC := S
5	JGE S	If ACC >= 0, PC := S
6	JNE S	If ACC != 0, PC := S
7	STP	Stop

* S denotes operand field * [S] denotes contents of cell with address S

In MU0 assembly, labels like ONE in the following example need not to be present but it is helpful to read the code. And labels are substituted to address and recorded to the symbol table in the first pass of the assembler.

<Example of MU0 Assembly Program>

```
      ; Sample code
 0)      LDA ONE ;1
 1)      ADD ONE ;2
 2)      ADD ONE ;3
 3)      ADD ONE ;4
 4)      ADD ONE ;5
 5)      ADD ONE ;6
 6)      ADD ONE ;7
 7)      ADD ONE ;8
 8)      ADD ONE ;9
 9)      ADD ONE ;10
 10)     STO RES ;store result to RES
 11) ONE:   1
 12) RES:   0
```

The above example shows that the program calculates 10 by adding one to ACC ten times and then it is stored in memory. Codes that start with “;” are a comment and don’t affect the program.

b) 2-Pass Assembler

Assembler is classified by the number of passes over the source code before creating the machine code from it. 1-pass assembler passes over the source code once, in the same pass collecting the labels, resolving future references and doing the actual assembly. Whereas, a 2-pass assembler creates a table with all unresolved symbols in the first pass and then uses the 2nd pass to resolve these addresses.

The following example shows the first & second passes of the two-pass assembler.

```
   ; test.asm
 0)  LOOP: LDA COUNT
 1)      SUB MAX
 2)      JGE END
 3)      LDA SUM
 4)      ADD COUNT
 5)      STO SUM
 6)      LDA COUNT
 7)      ADD ONE
 8)      STO COUNT
 9)      JMP LOOP
 10) END:   STP
 11) MAX:   100
 12) COUNT:   1
 13) SUM:   0
 14)     ONE:   1
```

Name	Address
LOOP	0
END	10
MAX	11
COUNT	12
SUM	13
ONE	14

(a) <First pass of 2-pass assembler>

```
  ; test.asm
0)  LOOP: LDA COUNT
1)     SUB MAX
2)     JGE END
3)     LDA SUM
4)     ADD COUNT
5)     STO SUM
6)     LDA COUNT
7)     ADD ONE
8)     STO COUNT
9)     JMP LOOP
10)  END: STP
11)  MAX: 101
12)  COUNT: 1
13)  SUM: 0
14)    ONE:   1
```

```
0)   0000000000001100
1)   0011000000001011
2)   0101000000001010
3)   0000000000001101
4)   0010000000001100
5)   0001000000001101
6)   0000000000001100
7)   0010000000001110
8)   0001000000001100
9)   0100000000000000
10)  0111000000000000
11)  0000000001100101
12)  0000000000000001
13)  0000000000000000
14)  0000000000000001
```

(a) <Second pass of 2-pass assembler>

3. Requirement

In this project, students are required to implement two programs; mu0 assembler and simulator.

a) Assembler
The input of the MU0 assembler is an assembly program. The above assembly code can be one of the examples. The output of the MU0 assembler is an object code which is shown above example. This object code consists of only 0 and 1. When the assembler receives input files, first it checks all of the labels with their address. Those labels and addresses are stored in Symbol Table. And then one line of instruction is translated to one line of binary code and written in the output file. Both input and output are handled as a file. Also after running the assembler, the symbol table should be shown on the screen. If there are errors like unknown instructions, the assembler shows an error message on the screen and has no output file.

Error Messages	Description
Unknown Instruction	Unknown instruction like "MUL" exists in an assembly code
Out of Memory	The size of an assembly code is larger than the total memory size (4K words)
Undefined Symbol	An instruction uses a certain label but there isn't the address reference of the label.
Other Error	Any kinds of errors except the above errors.

<Running Scenario>
(1) Success Scenario

```
~$ ./assembler
Enter the assembly code file name: test.asm
Running...
Symbol  table  Table
LOOP      :      0
END       :      10
MAX       :      11
COUNT     :      12
```

```
SUM      :      13
ONE      :      14
Output is test.obj
End of Program
~$ls
assembler.c   assembler   test.asm    test.obj
~$
```

(2) Failed Scenario

```
~$ ./assembler
Enter the assembly code file name: error.asm
Running...
ERROR!!
Unknown Instruction
End of Program
~$ ls
assembler.c   assembler   error.asm
~$
```

b) Simulator
The input of the MU0 simulator is an object code from the MU0 assembler. You can see one of an example of object codes in section 2.2. The output of the MU0 simulator is values of memory and ACC. Input is handled as a file and output is shown on the screen. If there are errors like unknown instructions, the MU0 simulator shows an error message on the screen.

Error Messages	Description
Unknown Instruction	Unknown instruction like “1111” exists in an object code
Out of Memory	The size of an assembly code is larger than the total memory size (4K words)
Other Error	Any kinds of errors except the above errors.

<**Running Scenario**>
(1) Success Scenario

```
~$ ./simulator
Enter the assembly code file name: test.obj
Running...
Memory
[1]    12
[2]    12299
[3]    20490
[4]    13
[5]    8204
[6]    4109
[7]    12
[8]    8206
[9]    4108
[10]   16384
[11]   8672
[12]   100
```

```
[13]   101
[14]   5050
[15]   1
ACC    0
ENd of Program
~$
```

(2) Failed Scenario

```
~$ ./simulator
Enter the assembly code file name: error.obj
Running...
ERROR!!
Unknown Instruction
End of Program
~$
```

4. Todo

In the homework, Try the three tasks below.

a) Understand of MU0
You understand about MU0 (a simple comupter) by the following reference page.
http://www.cs.man.ac.uk/~pjj/cs1001/arch/node1.html

b) Write the 2-Pass Assembler code
As a result, you should create a 2-pass assembler and generate two files(asm file and obj file). You need to initialize the running total to 0. This machine uses both explicit locations (10 and 11) and named locations (again, more and zero). An assembler will translate names to explicit location numbers as it converts the program into the representation stored in memory. In fact, it is normally preferable to use named locations all the time when writing programs.

c) Run the Test Code
We provided one example of MU0 in this document, but it isn't enough to test your program. You need to make one or more MU0 assembly codes for testing. Although you can find many examples on the web, we suggest you make your own code. It will help understand the MU0 assembly code.

5. Submit and Grade

Students do not submit for this assignment, and we do not evaluate it. But be sure to try this yourself.