SPIM for Windows—A Tutorial

Salah A. Almajdoub
smajdoub@eng.uob.bh
University of Bahrain
Electrical Engineering Department

Introduction

This is a tutorial to SPIM for windows; a simulator for MIPS 2000 machines. This tutorial teaches you how to load and run a simple assembly language program. Breakpoints and step execution techniques used for debugging are also presented.

A Detailed Example

Start SPIM and open the following file; **add.s**. This file contains a simple assembly program that calculates the sum of an array matrix N and stores the sum at a memory location `Result`. The array N is stored at memory location `Nstart` and the size of the array is stored at `Size`.

```
#####################################################################
# add.s
# addition of an array
#
#####################################################################

     .text
     .align 2
     .globl main
main:
     lw   $a0, Size       # read the size of array
     li   $a1, 0          # index i
     li   $a2, 0          # a2 contains the sum
     li   $t2, 4          # t2 contains the constant 4

loop: mul  $t1, $a1, $t2   # t1 gets i * 4
     lw   $a3, Nstart($t1)# a3 = N[i]
     add  $a2, $a2, $a3   # sum = sum + N[i]
     add  $a1, $a1, 1     # i = i + 1
     beq  $a1, $a0, STOR  # go to STOR if finished
     j    loop


STOR: sw   $a2, Result     # store the sum at Result

     .data
     .align 2
Nstart:    .word 8, 25, -5, 55, 33, 12, -78
Size:      .word 7
Result:    .word 0
```

add.s: A program to calculate the sum of elements in an array

SPIM creates four windows; text segment, data segment, massages and registers windows. The text segment window shows the add program as loaded to the memory. The first column gives the memory location of each instruction and the second column gives the instruction encoding in hexadecimal. The actual instructions follows the instruction encoding. The add program contains several psuedinstructions but SPIM can only load real instructions to the memory. Hence, every line contains the real instruction as loaded to the memory followed by the original instruction in the assembly program file. For example, load immediate instruction `li $a1, 0` is actually a pseudoinstruction and its implementations is `ori $5, $0, 0`.

Pseudoinstructions can be represented by up to three actual instructions. Notice that the first instruction

```
lw   $a0, Size
```

is represented by two instructions

```
lui  $1, 4097
lw   $4, 28($1)
```

These two instructions work together to read the size of the array N from address 0x1001001c. The first instruction loads register \$1 with 0x10010000 while the second instruction calculates the address of Size by adding 28 to register \$1 producing the address 0x1001001c. Then the content of this address is loaded to register \$4 (\$a0).

```
[0x00400000]   0x3c011001  lui $1, 4097 [Size]; 11: lw  $a0, Size # read the size of array
[0x00400004]   0x8c24001c  lw $4, 28($1) [Size]
[0x00400008]   0x34050000  ori $5, $0, 0     ; 12: li   $a1, 0     # index i
[0x0040000c]   0x34060000  ori $6, $0, 0     ; 13: li   $a2, 0     # a2 contains the sum
[0x00400010]   0x340a0004  ori $10, $0, 4    ; 14: li   $t2, 4  # t2 contains the constant 4
[0x00400014]   0x00aa0018  mult $5, $10      ; 16: mul  $t1, $a1, $t2
[0x00400018]   0x00004812  mflo $9
[0x0040001c]   0x3c011001  lui $1, 4097 [Nstart] ; 17: lw $a3, Nstart($t1)
[0x00400020]   0x00290821  addu $1, $1, $9
[0x00400024]   0x8c270000  lw $7, 0($1) [Nstart]
[0x00400028]   0x00c73020  add $6, $6, $7        ; 18: add $a2, $a2, $a3
[0x0040002c]   0x20a50001  addi $5, $5, 1    ; 19: add $a1, $a1, 1        # i = i + 1
[0x00400030]   0x10a40002  beq $5, $4, 8 [STOR-0x00400030] ; 20: beq        $a1, $a0, STOR
[0x00400034]   0x08100005  j 0x00400014 [loop]             ; 21: j        loop
[0x00400038]   0x3c011001  lui $1, 4097 [Result] ; 24: sw a2, Result # store the sum at Result
[0x0040003c]   0xac260020  sw $6, 32($1) [Result]
```

Text Segment Window Content

The Data Segment window display shows the data loaded into your program's memory. Locations 0x10010000 to 0x10010018 should include the elements of the N array $\{8, 25, -5, 55, 33, 12, -78\}$. The values appear in hexadecimal. Location 0x1001001c, which contains $Size$, is the last data item in the line that starts with address 0x10010010. Your Data Segment window should display 0x0000007 at the memory location 0x1001001c indicating that $Size$ is initialized to 7. Remember that each word needs the addresses of four bytes.

```
[0x10010000]        0x00000008  0x00000019  0xfffffffb  0x00000037
[0x10010010]        0x00000021  0x0000000c  0xfffffff2  0x00000007
```

Part of Data Segment Window Content

To run this program you need to select (simulator → go) from SPIM menu or press F5. This should pop up a window that asks for the starting address. Type 0x00400000 which is the address of the first instruction in memory then click OK. The add program should run and

then the memory location 0x10010020 should contain 0x00000032 i.e. the Result location is set to 50 which is the sum of the array elements.

Usually programmers don't start with correct programs so they study the behavior of their programs using a process called debugging. Two main tools are used for debugging programs; breakpoints and step by step execution. Select (simulator → breakpoints) and write the address 0x0040000c then click add and close. This should add a break point at the instruction

```
li $a2, 0
```

Select (simulator → go) to run the program. The program should stop at the breakpoint. Select No when your asked if you want to continue execution. Now you should try step execution technique. In this technique you run the program step by step. Select (simulator → single step) or press F10. Notice how one line of code gets executed. Press F10 again and watch the changes in the registers in the registers window. Repeat pressing F10 until you finish running the whole program or you can select (simulator → continue) to continue the execution till the end of the program.

The log file is a text file that contains information about how the program was loaded and run. Also the log file records the content of the registers, text segment and data segment. To generate a log file select (file → save log file). Then provide a name for this file e.g. addrun.log.