CS 241 — Spring 2020 — Assignment 2

In this assignment, you will use assembly language to write somewhat more complicated programs than in Assignment 1.

Reminder: For this and future assignments, be sure to run the command `source /u/cs241/setup` to gain access to the CS 241 tools.

Marmoset Notes

For each question, Marmoset has public and release tests, and often has secret tests as well. The weights assigned to each question are given on the assignment.

When you release test a submission, Marmoset will only report on public and release tests. No information about secret tests will be visible until after the assignment deadline.

This means the number of marks you see on Marmoset (prior to the assignment deadline) will not necessarily match the number of marks shown on the assignment page.

Part I. Introduction to MIPS Assembly Language

In this part of the assignment you will use the `mips.twoints` tool that you used in Assignment 1.

In addition, we have provided an assembler, `cs241.binasm`, which accepts assembly language on standard input, and outputs on standard output a MIPS machine language program suitable for use with `mips.twoints` (and `mips.array` introduced in Part II.) Detailed documentation about MIPS Assembly Language is available at http://www.student.cs.uwaterloo.ca/~cs241/mips/mipsasm.html. The program `cs241.binasm` will be available after the deadline for Assignment 1 has passed.

For problems 1 and 2, submit only the MIPS assembly language version of your solution. Each problem specifies a filename for the file that you will submit. You must use the specified filename for the testing scripts to work correctly.

Write MIPS assembly language programs that accomplish the following tasks:

Problem 1 — 6 marks of 61 (filename: a2p1.asm)

Registers 1 and 2 hold 32-bit integers (in two's complement notation). Place the maximum of these integers in register 3, and return.

Example:

```
Enter value for register 1: -1
Enter value for register 2: 3
Running MIPS program.
MIPS program completed normally.
$01 = 0xffffffff   $02 = 0x00000003   $03 = 0x00000003   $04 = 0x00000000
...
```

Click here to go back to the top of the page.

Problem 2 — 6 marks of 61 (filename: a2p2.asm)

Registers 1 and 2 hold 32-bit integers (in unsigned integer notation). Place the maximum of these integers in register 3, and return.

Example:

```
Enter value for register 1: 1
Enter value for register 2: 3
Running MIPS program.
MIPS program completed normally.
$01 = 0x00000001   $02 = 0x00000003   $03 = 0x00000003   $04 = 0x00000000
...
```

Click here to go back to the top of the page.

Part II. MIPS Assembly Language — Arrays

In this part of the assignment, you will use the assembler `cs241.binasm` and `mips.array` which is a tool similar to `mips.twoints`.

The `mips.array` tool loads a MIPS machine language program from a file into memory starting at location 0. It then reads an integer n (the length of an array), then reads n more integers. The length n is stored in register 2. The subsequent n integers are stored in an array of consecutive 32-bit integers in memory, and the address of the beginning of this array is stored in register 1. Your program is then executed. When your program returns, the tool prints the values of all the registers and exits. Run this tool using the command `mips.array <mycode>`, replacing `<mycode>` with the name of a file containing your machine code.

For problem 3, submit a file containing the MIPS assembly language version of your solution and a file containing the assembled MIPS machine language version of your solution. For all problems after problem 3, submit only the MIPS assembly language version of your solution. Each problem specifies a filename for the file that you will submit. You must use the specified filename for the testing scripts to work correctly.

Write MIPS assembly language programs that accomplish the following tasks:

Problem 3 — 8 marks of 61

Register 1 holds the address of the beginning of an array of 32-bit integers. Register 2 holds the number of elements in the array. If the array is empty place the value -1 in register 3. Otherwise copy the **last** element of the array into register 3, and return.

Example:

```
Enter length of array: 2
Enter array element 0: 1
Enter array element 1: 2
MIPS program completed normally.
$01 = 0x0000003c   $02 = 0x00000002   $03 = 0x00000002   $04 = 0x00000044
...
```

Part A: (filename: `a2p3.asm`)

Submit the MIPS assembly language version of your solution.

Part B: (filename: `a2p3.mips`)

Submit the MIPS machine language produced by `cs241.binasm` from the assembly language version of your solution.

Click here to go back to the top of the page.

Problem 4 — 8 marks of 61 (filename: `a2p4.asm`)

Register 1 holds the address of the beginning of an array of 32-bit two's complement integers. Register 2 holds the number of elements in the array. Determine the **maximum** of all the elements of the array, write it into register 3, and return. Assume the array is not empty.

Example:

```
Enter length of array: 5
Enter array element 0: 32
Enter array element 1: 24
Enter array element 2: 54
Enter array element 3: 9
Enter array element 4: 0
MIPS program completed normally.
$01 = 0x00000060   $02 = 0x00000005   $03 = 0x00000036   $04 = 0xffffffff

Note: Register 3 contains 0x36 which is 54 in hexadecimal.
```

Click here to go back to the top of the page.

Problem 5 — 8 marks of 61 (filename: `a2p5.asm`)

Register 1 holds the address of the beginning of an array of 32-bit integers, each representing a character. The integer zero represents a space, and each integer i (1 <= i <= 26) represents the i'th letter of the uppercase alphabet. Register 2 holds the number of elements in the array (can be empty). Your program should output the uppercase characters represented by the integers in the array, and return. The MIPS system allows you to output one character at a time, by storing its ASCII value into the special memory location ffff000c (hex) (Hint: you may find the command `man ascii` helpful.)

Example:

```
Enter length of array: 11
Enter array element 0: 8
Enter array element 1: 5
Enter array element 2: 12
Enter array element 3: 12
Enter array element 4: 15
Enter array element 5: 0
Enter array element 6: 23
Enter array element 7: 15
Enter array element 8: 18
Enter array element 9: 12
Enter array element 10: 4
HELLO WORLDMIPS program completed normally.
$01 = 0x0000006c   $02 = 0x0000000b   $03 = 0x00000000   $04 = 0x00000004
...
```

The above is what you might see if both standard output (where "HELLO WORLD" is sent) and standard error (where the MIPS register contents are sent) are sent to the same place. You might want to redirect them to different places when testing.

Click here to go back to the top of the page.

Problem 6 — 10 marks of 61 (filename: `a2p6.asm`)

Register 1 holds a 32-bit integer (in two's complement notation). Your program should format this integer in base 10, print it, then print a newline character, and return.

Example:

```
Enter value for register 1: 5
Enter value for register 2: 3
Running MIPS program.
5
$01 ......
```

Another Example (not a public test):

```
Enter value for register 1: -55
Enter value for register 2: 3
Running MIPS program.
-55
$01 ......
```

Click here to go back to the top of the page.

Problem 7 — 7 marks of 61

Part A: (filename: `print.asm`)

Encapsulate your program from Problem 6 in a procedure with the label `print`. Your procedure should format the integer in register 1 in base 10, print it, print a newline character, and return. When your procedure returns, all registers should have the same values that they had when the procedure was called. For example, the procedure could be called by the following program, which uses the procedure to print the number 12345:

```
; push $31 on stack
sw $31, -4($30)
lis $31
.word -4
add $30, $30, $31
```

```
lis $1
.word 12345
lis $2
.word print
jalr $2 ; call procedure

; pop $31 from stack
lis $31
.word 4
add $30, $30, $31
lw $31, -4($30)

jr $31

; your code for print procedure should be added here
```

Submit a file called `print.asm` containing the MIPS assembly language version of only your procedure. Although you will need to combine your procedure with a main program such as the one above to test it, submit only the procedure to Marmoset.

Click here to go back to the top of the page.

Part B: (filename: `a2p7.asm`)

Write a program that will be executed with an array of 32-bit integers (with register 1 holding the address of the beginning of the array, and register 2 holding the number of elements in the array, as before). Your program should call your integer-printing procedure from Part A for each integer in the array in turn, to print all the integers, and return. The program should simply return if the array is of length 0.

Example:

```
Enter length of array: 3
Enter array element 0: 1
Enter array element 1: 2
Enter array element 2: 3
1
2
3
MIPS program completed normally.
...
```

Submit a file called `a2p7.asm` containing the MIPS assembly language version of only your main program, but not the `print` procedure. Although you will need to combine your main program with the procedure to test it, submit only the main program to Marmoset.

Click here to go back to the top of the page.

Problem 8 — 8 marks of 61 (filename: `a2p8.asm`)

Your task is to determine the height of a binary tree. Assume every tree will contain at least one node and hence have height at least one.

The tree will be encoded in an array of 32-bit integers. Each node of the tree is encoded in three consecutive elements (words) of the array: a two's complement integer stored at the node, the node's left child, and the node's right child. Each child is specified as the array index of the first element of the child node. The integer

-1 indicates that a node does not have a left or right child. For example, the following tree:

```
     77
    /  \
   22   -8
       /  \
     -36   999
```

could be encoded by following array:

```
A[0] = 77
A[1] = 3
A[2] = 6
A[3] = 22
A[4] = -1
A[5] = -1
A[6] = -8
A[7] = 9
A[8] = 12
A[9] = -36
A[10] = -1
A[11] = -1
A[12] = 999
A[13] = -1
A[14] = -1
```

in which the root is encoded by the elements A[0], A[1] and A[2], the root's left child is encoded by the elements A[3], A[4] and A[5], the root's right child is encoded by the elements A[6], A[7] and A[8], the root's left-most grandchild is encoded by the elements A[9], A[10] and A[11], and the root's right-most grandchild is encoded by the elements A[12], A[13] and A[14]. This tree has height 3.

Register 1 holds the address of the beginning of the array, and register 2 holds the number of elements in the array. Determine the height of the tree, store it in register 3, and return. Your solution should only read, but NOT MODIFY, the tree in memory.

Example:

```
Enter length of array: 15
Enter array element 0: 77
Enter array element 1: 3
Enter array element 2: 6
Enter array element 3: 22
Enter array element 4: -1
Enter array element 5: -1
Enter array element 6: -8
Enter array element 7: 9
Enter array element 8: 12
Enter array element 9: -36
Enter array element 10: -1
Enter array element 11: -1
Enter array element 12: 999
Enter array element 13: -1
Enter array element 14: -1
MIPS program completed normally.
$01 = 0x000000c4   $02 = 0x0000000f   $03 = 0x00000003   $04 = 0x00000000
...
```

Click here to go back to the top of the page.

Bonus Problem — 3% bonus on Assignment 2 grade (filename: `a2bonus.asm`)

Register 1 holds the address of the beginning of an array of 32-bit integers. Register 2 holds the number of elements in the array. Determine the **median** of all the elements of the array, write it into register 3, and return. Assume the length of the array is a positive odd number. Your program must run in O(n log(n)) time, where n is the length of the input array. Include comments explaining the algorithm that you are using.

Example:

```
Enter length of array: 3
Enter array element 0: 1
Enter array element 1: 3
Enter array element 2: 2
MIPS program completed normally.
$01 = 0x0000021c   $02 = 0x00000003   $03 = 0x00000002   $04 = 0x00000000
...
```

Click here to go back to the top of the page.

Extra bonus (no marks, nothing to hand in, just something to think about): Can the median be found in strictly better than O(n log(n)) time?