CS 241 — Spring 2020 — Assignment 1

Assignments for CS 241

Assignment 1

Friday, May 22nd at 5:00pm

Assignment 2 →

~~Friday, May 29th at 5:00pm~~
Sunday, May 31st at 5:00 pm

P1 • P2 • P3 • P4 • P5 • P6

In this assignment, you will begin writing very simple programs in MIPS machine language on the CSCF Linux environment.

You must prepare and test your solutions using tools available on the CSCF Student Linux Computing Environment. You must submit your solutions to the Marmoset automatic grading system available at `https://marmoset.student.cs.uwaterloo.ca`, which will run them on a number of test inputs and grade them automatically. You may submit your solutions as many times as you wish prior to the submission deadline (though the number of test runs is limited; see below). Your mark is determined entirely by the set of test inputs for which your best submission generates the correct answer.

You must test your submissions prior to submitting them to Marmoset. You should not rely on Marmoset as a means of submission testing.

Marmoset will test your assignment using two kinds of tests: public tests and release tests. For each problem, there is one public test input, which is included as an example in this document at the end of each problem (except Problem 6). Marmoset allows you to view the output of your code when executed with the public test input. Once your code passes the public test, you may ask Marmoset to run it on the remaining release tests. Marmoset will not show you the output from these tests, but it will tell you how many tests you passed, and the names of the first two tests that you failed. Marmoset also provides the test input for some release tests. You can use this information to fix your code and resubmit. For some problems, the test input and output of certain "blind" tests are never released, either by Marmoset or by course staff.

To encourage you to start the assignment early, the number of release test runs that you may request for each problem is limited to three in every 12-hour period. Marmoset gives you three release test tokens for each problem. Each test run uses up one token. The token is returned to you 12 hours after the test run request.

You can use the `marmoset_submit` command to submit to Marmoset directly from the command line, avoiding the need to copy files back and forth between your computer and the Linux environment.

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

Part I. Creating binary files with `cs241.wordasm`

We have provided the tool `cs241.wordasm` that may be used to create a file whose binary content is specified using hexadecimal notation. `cs241.wordasm` reads the hexadecimal representation of several 32-bit words on standard input. On standard output, it outputs a file containing, for each of the input words, the four bytes containing the 32 bits represented by the word (8 bits per byte). On each line of input, any characters after a semicolon (;) are assumed to be comments and ignored. Example:

```
% cat > input
```

```
.word 0x43533234  ; C(43) S(53) 2(32) 4(34)
.word 0x3120726f  ; 1(31) space(20) r(72) o(6f)
.word 0x636b730a  ; c(63) k(6b) s(73) newline(0a)
% cs241.wordasm < input
CS241 rocks
% cs241.wordasm < input > output
% xxd < output
00000000: 4353 3234 3120 726f 636b 730a            CS241 rocks.
```

Note that the `xxd` Linux command shows the contents of the file both as hexadecimal numbers and as ASCII characters.

Problem 1 — 8 marks of 46 (filename: `helloworld.hex`)

Write a hexadecimal representation (using the .word notation) of a file with contents:

```
Hello
from Linux!!!
```

You may find it useful to use the LINUX command `man ascii` to get an ASCII code chart. Before submitting to Marmoset, run the `cs241.wordasm` tool with your solution as standard input to check that your output is correct.

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

Part II. MIPS Machine Language Programming

The `mips.twoints` tool loads a MIPS machine language program from a file into memory starting at location 0. It then reads two integers, stores them into registers 1 and 2, runs your program, and, when your program returns to the instruction whose address is stored in register 31, prints the values of all the registers and exits. Run this tool using the command `mips.twoints` ***`mycode.mips`***, replacing ***`mycode.mips`*** with the name of a file containing your machine code.

Since MIPS machine language is encoded in binary, not text, you cannot create it directly using an editor like vim. You must specify your machine language program in hexadecimal and use `cs241.wordasm` to translate it to (binary) machine language.

For each of the following problems, create the hexadecimal representation of a MIPS machine language program that solves the problem. Test your program using `cs241.wordasm` and `mips.twoints` as follows:

```
% vim mycode.hex
% cs241.wordasm < mycode.hex > mycode.mips
% mips.twoints mycode.mips
Enter value for register 1: 1
Enter value for register 2: 2
Running MIPS program.
...
```

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.

Scratch registers used to hold temporary results may be used and do not need to be set back to 0.

Problem 2 — 8 marks of 46 (filename: `a1p2.hex`)

Write the hexadecimal notation for a MIPS machine language program that returns to the address saved in register 31. This is the only thing your program should do.

Example of running the program:

```
Enter value for register 1: 1
Enter value for register 2: 2
Running MIPS program.
MIPS program completed normally.
...
```

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

Problem 3 — 8 marks of 46 (filename: `a1p3.hex`)

Write the hexadecimal notation for a MIPS machine language program that copies the value in register 1 to register 3, then adds the values in register 1 and 3 placing the result in register 4 and then returns.

Example of running the program:

```
Enter value for register 1: 2
Enter value for register 2: 3
Running MIPS program.
MIPS program completed normally.
$01 = 0x00000002   $02 = 0x00000003   $03 = 0x00000002   $04 = 0x00000004
...
```

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

Problem 4 — 8 marks of 46 (filename: `a1p4.hex`)

Write the hexadecimal notation for a MIPS machine language program that determines the minimum of the two's complement signed integers in registers 1 and 2, places it in register 3, and returns.

Example of running the program:

```
Enter value for register 1: 3
Enter value for register 2: 5
Running MIPS program.
MIPS program completed normally.
$01 = 0x00000003   $02 = 0x00000005   $03 = 0x00000003   $04 = 0x00000001
...
```

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

Problem 5 — 7 marks of 46 (filename: `a1p5.hex`)

Write the hexadecimal notation for a MIPS machine language program that adds 42 to the value in register 2 placing the result in register 3 and then returns.

Example of running the program:

```
Enter value for register 1: 2
Enter value for register 2: 3
Running MIPS program.
MIPS program completed normally.
$01 = 0x00000002   $02 = 0x00000003   $03 = 0x0000002D ...
```

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

Problem 6 — 7 marks of 46 (filename: `a1p6.hex`)

Write the hexadecimal notation for a MIPS machine language program that interprets the value in register 1 as the address of a word in memory, and places the address of the following word in memory in register 3 and then returns.

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