COSC 2440 – Computer Organization and Architecture – Spring 2020 - Kevin B Long

# 

# Homework #3

Due 11:59pm, Sunday, Mar 1, 2019

Multiple submissions accepted, last one graded.

101 points total.

Name: Xena Toumajian

PeopleSoft ID: 1662518

**2.1** (5 pts) [5] <COD §2.2> For the following C statement, what is the corresponding 3-line MIPS assembly program? One special request: do not overwrite any of the values in the registers that are part of your equation, except for the result. However, you may use $t4 as a temporary register if you need to store an intermediate value.

x = a \* (b ÷ (c + d);

Let a, b, c, and d be in $t0 through $t3, and x in $s0

ADD $s0, $t2, $t3

DIV $s0, $t1, $s0

MUL $s0, $t0, $s0

**2.2** (5 pts) [5] <COD §2.2> For the following MIPS assembly instructions above, what is a corresponding C statement? Obviously I’m not giving you real register names, just go with the variable names shown.

sub a, c, a

add b, a, d

mul c, b, a

a= c-a;

b= (c-a)+d;

c= ((c-a)+d)\*(c-a);

**2.A** (25 pts) Navigating the Green Card

In Figure 2.1 in our primary textbook is a table of MIPS instructions. It is a summary of the information found on the green card (the file *mips\_reference\_data (Greeen Card).pdf* in the General folder on the class drive). Focus your attention on the first page of that card and answer the following questions. Not all questions have answers, some are tricks.

1. (2 pts) What is the instruction format for “Or”? \_\_\_\_R\_
2. (2 pts) What about “Or Immediate”? I
3. (2 pts) It’s common sense that OR is going to logically look bitwise at inputs, not try and treat them as 2’s complement numbers to be added. But “Add Immediate” does treat the immediate value as a 2’s complement number. Look at the Operation equation for “OR Immediate”. What about that equation makes it obvious that the system is not treating it as a 2’s complement number? Compare it to the equation for “Add Immediate” if you don’t see the answer at first.

The ZeroExtImm which inserts 0s to the empty bits. In twos compliment you would copy the most signifigant bit to the rest of the empty bits (SignExtImm).

1. (2 pts) What is the instruction format for Jump and link? J
2. (2 pts) What is the OPCODE for load Store conditional (in hex)? 38
3. (2 pts) What is the mnemonic for Load Byte Unsigned? lbu
4. (4 pts) What mnemonic does an OPCODE of “01001” represent? addiu

What instruction format would we use to interpret the rest of the bits? I

1. (2 pts) If you find the value 2 in bits 0-5 of an instruction, what instruction would you assume is being run?

☐ Jump ☐ srl ☐ any I-type instruction ☐ can’t tell

1. (2 pts) What’s the opcode for the “nop” (also called “no op”) command?

000000

1. (2 pts) Which commands may cause overflow exceptions (mnemonic, please)?

Add, addi, sub

1. (3 pts) Here is an English-language description of what one of the commands does that can be found on the first page of the MIPS reference data sheet (I promise):

Take the address of the instruction currently being run (where you yourself were found). Calculate the address of the instruction that is two instructions later and store it for safekeeping in the Return Address register.

Now calculate the address of the instruction that is one instruction later than your own. Then take the high 4 bits of that address, concatenate it with bits 0-25 of your original 32-bit instruction, and add 2 zeros to the end to make it a new 32-bit address. Then change the contents of the program counter to that number.

What is the name of this instruction (not the mnemonic)?

Jump and Link

**2.B** (15 pts) Go to <https://www.random.org/bytes/>. Generate 4 random bytes of hexadecimal, which represents 32 bits, the same size as a MIPS instruction.

Paste the hex string you used here:

< 36 73 59 0f>

Enter it without the spaces at <https://www.eg.bucknell.edu/~csci320/mips_web/> to see if it will convert to a valid instruction. If you don’t get one, try again until you do.

Copy and paste the results as text below, and then answer the questions that follow. Tip: if you convert what you paste to the Courier or Typewriter font and reduce the font size, the results will be more readable.

# < ORI $s3 $s3 0x590F>

### Binary: 00110110011100110101100100001111

### Hex: 0x3673590f

Example:

Hex: c1ff9950

## Result

## LL $ra 0x9950 $t7

### Binary: 11000001111111111001100101010000

### Hex: 0xc1ff9950

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 31 | 26 | 25 | 21 | 20 | 16 | 15 | 0 |
| LL | | $t7 | | $ra | | offset | |
| 110000 | | 01111 | | 11111 | | 1001100101010000 | |
| 6 | | 5 | | 5 | | 16 | |

1. What instruction did you create?

Mnemonic: ori

Name: Or immediate

1. Was the instruction on the MIPS Green Sheet from class? ☐ Yes ☐ No

If so, in which group of instructions was it found?

☐ Core instruction set

☐ Arithmetic core instruction set

☐ Pseudoinstruction set

1. Is it in the “mips-iv instruction reference.pdf” document (in Extras)? ☐ Yes ☐ No

If so, on what page? \_\_\_\_\_\_\_\_

1. According to the Bucknell web site, in what MIPS architecture was the instruction first introduced?

☐ MIPS I ☐ MIPS II ☐ MIPS III ☐ MIPS IV

1. What instruction format does your instruction use?

☐ R-type ☐ I-type ☐ J-type ☐ FR-type ☐ FI-type ☐ Other:\_\_\_\_\_\_\_

**2.C** (20 pts) Continue with the hex string from above. You will need to refer to the green sheet (see the Extras folder).

a. Use <https://www.binaryhexconverter.com/hex-to-binary-converter> to convert the 8-character hex string to binary. It will print its results with spaces every four bits. That will help us keep track of where we are. Copy and paste that 32-bit answer from the web as text (not as a photo) so that the TA can copy it and test your answers. LEAVE THE SPACES so we can read the string.

<0011 0110 0111 0011 0101 1001 0000 1111 >

b. Even if it is not true, let’s assume the instruction is an R-type instruction. Fill in the bits accordingly into the R-type format:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Opcode | rs | rt | rd | shamt | funct |
| 001101 | 10011 | 10011 | 01011 | 00100 | 001111 |

c. OK, if we wanted the instruction to actually be an “add” command, to what values would we need to change opcode and funct above?

Opcode: 000000

Funct: 100000

Indicate if you answered in ☐decimal, ☐hex, or ☐binary.

d. Replace xx, yy and zz with “rs”, “rt”, and “rd” to show how they’re used in a line of assembly language code in the proper order:

add rd, rs, rt

e. What math equation actually gets calculated by the above add command? If you think each pair of answers are the same, consider if it was a subtraction or division command instead. Thus there is only one right answer.

☐rd + rs = rt

☐rs + rd = rt

☐rs + rt = rd

☐rt + rs = rd

☐rd + rt = rs

☐rt + rd = rs

**3.A** (12 pts) IEEE-754   
Use <https://www.h-schmidt.net/FloatConverter/IEEE754.html> to solve the following problems single-precision IEEE-754-formatted numbers. For most, you will be asked to copy and paste the row of bits the tool displays to show your answer. In other words, this graphic: 

…but with the bits set with your answer.

1. (2 pts) First a question that requires no graphics: when you take a number like *a • 2b*, mathematically what do you replace b with before storing it in the exponent field?

That number plus the bias

1. (2 pts) First, show how the bits are set to represent the binary number which before normalization is 1100.10 \* 23. You’ll have to normalize it and then set the bits accordingly.

A close up of a logo

Description automatically generated

1. (2 pts) The decimal number -3.6909.

A close up of a logo

Description automatically generated

Can this number be precisely represented in IEEE-754? ☐Yes ☐No

1. (2 pts) The value -∞ (Hint: read the web page)

A close up of a logo

Description automatically generated

1. (2 pts) The value 0xc1300000:

A close up of a logo

Description automatically generated

1. (1 pt) What type of number is represented when you clear all of the exponent bits?

A denormalized number

1. (1 pt) What about if you set all exponent bits and set at least one mantissa bit?

Not a number

**3.B** (2 pts) What is the name for the ability to run a program written in an earlier ISA that was available on an old version of a chip and to do so successfully on a new version of the chip with a different ISA?

Single instruction multiple data

What is the name of the most family instruction set architecture (three characters) that exhibits this technique?

X86

**3.C** (15 pts) <§3.5> Floating-point math

1. (1 pt) You are asked to normalize 1100.1101 x 28. Keeping all the bits, what do you get?

1.1001101 x 2^11

1. (1 pts) If you are told to restrict your answer to a total of 5 bits, and you round up (away from zero), what do you get?

1.1001 x 2^11

1. (7 pts) If you are adding two numbers: 1.1001011x220 and 1.0111010x222, what’s the correct sequence of steps? Number them 1-7. If there are two steps that seem to not matter in which order you do them, just pick one. Both will be counted as correct.

|  |  |
| --- | --- |
| 4 | Check for overflow |
| 7 | Round the result |
| 2 | Add the significands together |
| 6 | Renormalize if needed |
| 1 | Shift the significands to align radix points |
| 3 | Normalize the initial sum |
| 5 | Check for underflow |

1. (1 pts) What is the answer you get following your sequence? In other words, what’s the sum?

111.01101\*2^20

1. (2 pts) You want to multiply 1.0110110 x 215 and 1.0010000 x 2-3.

What’s the decimal value of both inputs?

46592 and 0.140625

1. (1 pt) Following the process in slide 38 of chapter 3 (see the slides folder), what’s the result? Use the same number of bits in your answer as were used in the sources.

1.10011001 x10^12

1. (1 pt) Here’s a table of negative powers of 2:

|  |  |  |  |
| --- | --- | --- | --- |
| Pwr of 2 | Frac | Dec | Bin |
| -1 | ½ | 0.5 | .1 |
| -2 | ¼ | 0.25 | .01 |
| -3 | 1/8 | 0.125 | .001 |
| -4 | 1/16 | 0.0625 | .0001 |
| -5 | 1/32 | 0.03125 | .00001 |

Use it to calculate the binary value of 1.4375. Circle/mark/highlight the binary values you added to your answer from the table.

1= .1+.1 =1.0

.4375= .1-.0001=.0111

1.0111

1. (1 pt) If you are given the 32-bit single-precision IEEE 754 format floating point number expressed as a hexadecimal string:

0x41300000

Which in binary is:

0100 0001 0011 0000 0000 0000 0000 0000

Mark the part of this string that is used as the sign bit by highlighting it in yellow or marking it with an “S”.

Highlight in blue or otherwise mark with an “E” the part used as the exponent.

Mark the mantissa bits by highlighting in green or otherwise marking with an “M”