HOMEWORK#1 DUE: 02/08/2021

MIPS assembly (100pts)

Since computer hardware can only communicate in 0's and 1's, our programs written in MIPS must be translated into machine code containing only 0's and 1's so that they can be executed. In this lab, we will review the conversions between MIPS and machine code.
We will use MARS (our beloved MIPS simulator!) throughout this assignment. Feel free to read the attached documents (MARS Tutorial.pdf, MARS features.pdf) if you have not used MARS before or forgot about how to use it.

If you did not learn MIPS when you learned assembly language, please feel free to group with other students who have learned MIPS. You will pick it up easily with their help.
(Exercise) MIPS ↔ Machine Code

Find a classmate to work with and perform the following tasks (You are allowed to form a group of three):
1. Record the name(s) of your partner(s).

Luis Garcia-Velasquez
Josue Gialis
Christian Baronia

2. Download "MIPS Reference Data_full.pdf" from CatCourse. We will need to refer to this sheet in order to complete all the exercises in this lab.

3. Load proc1.s in MARS and study the code.

4. After assembling the program, study the Text Segment window and see how your source code is translated into True Assembly Language (Basic) as well as machine code (Code).

5. In true assembly language, every single instruction can be translated into a machine instruction. How many bits does a machine instruction contain?

32 Bits

6. To utilize the limited number of bits efficiently, all machine instructions are categorized into different types (or formats). How many types are there? What are they? Give 2 operations for each type as examples.

- I
  - lw $2, 0($3)
  - sw $2, 0($3)
- J
  - j 0x00
  - j 0x01
- R
  - slt $1, $2, $3
  - sltu $1, $2, $3

LUIS

7. Now, locate the instruction in line #14 of proc1.s. Let's translate this instruction into machine code.

a. What instruction type is this?

R-type

How many fields does this type of instruction have?

6 fields

What are the names of these fields?

The fields are opcode, rs, rt, rd, shamt, funct

b. Refer to the MIPS sheet, what is the value of the opcode of this instruction in Hex?

0

What register is rs?

Rs is the first source register, $s1 in this case

What is the value of this register in Hex?

Its Hex value is 0x00000005

What register is rt?

Rt is the second source register, $s0 in this case

What is the value of this register in Hex?

Its Hex value is 0x00000019

What register is rd?

Rd is the destination register, $t0 in this case

What is the value of this register in Hex?

Its Hex value is 0x00000001

What is the value of the funct field of this instruction in Hex?

0x0000002A

c. Construct the machine code of line #14 using the values obtained from part b. Write your answer in both binary and Hex formats. You can verify your answer with the Code column in Text Segment window.

0x0230402A

000000 10001 10000 01000 00000 101010

Christian

8. Now, let's convert a machine code to a MIPS instruction. Locate address 0x00400024 from the Text Segment window.

a. What is the machine code at this address in Hex? Convert this code into binary.

b. From the binary version of this machine code.

What is the instruction type?

How can you tell?

How many fields are there in this instruction type?

What are the names of these fields?

c. According to the binary machine code, what is the value of each field in Hex?

d. Refer to the MISP sheet,

what operation is this instruction?

How can you tell?

What is the mapping of the registers being used in this instruction?

e. What is the final MIPS instruction?

Is it the same as the Source column in the Text Segment window?

A. 0x34240000 --> 001101 00001 00100 0000000000000000
B. Opcode != 0 && Opcode > 3 I-Type(4 Fields): Opcode(6), Rs(5), Rt(5), Imm(16)
C. Field Values
    a. Opcode: 001101 → 0xD
    b. Rs: 00001 → 0x1
    c. Rt: 00100 → 0x4
    d. Imm: 0000 0000 0000 0000 → 0x0000
D. Operation: ori, since opcode= 0xD (Look up in table)
    a. Rs → $1
    b. Rt → $4
    c. Imm → 0x0
E. "ori $4, $1, 0x00000000", Same as Source Column

JOSUE

9. Now, let's take a look at line #17 of proc1.s.

a. What format is this instruction?

I-format

b. What are the values of opcode, rs, and rt of this instruction in hex?

opcode 0x00000005
Rs ($t0) value 0x00000001
Rt ($zero) value always 0 (0x00000000)

c. What is the name of the target label if it takes the branch? What is the address of this label in hex? (Hint: you can find it in the Text Segment window.)

ANSWER:

1. Less
2. 0x0040001c

d. So, do we put this address as the value of the immediate field of the instruction? Why?

No you would not put the address
Of the label in the immediate field.
This is because we do not want to
Branch to the absolute address, rather
We want to branch to N lines above or below. Although,
If at compile time of our program we knew the address beforehand
Then we could use it in the immediate field.

e. How do we find the value of the immediate field? What is this value?

ANSWER: The value of the immediate field can be found by counting every line after the line in execution, until the label "leeq" and then multiplying that number by 4.
In this case, the value is 16

f. What is the machine code of this instruction in binary and hex formats? Does your answer match the Code column in the Text Segment window?

ANSWER:

000101 01000 00000 0000000000000001
0x15000001
Yes, my answer does match.

Christian

10. Finally, let's convert the j instruction in line #20.

a. What format is this instruction? How many fields are there in this format?
b. What is the opcode of this instruction in hex?
c. What label and address does this instruction jump to?
d. How many bits can you use in the address field of the instruction? How can we "squeeze" the address into this field? What are the reasons behind this approach? What is the value of the address field in binary?
e. What is the machine code of this instruction in binary and hex? Is it the same as what's in the Code column of the Text Segment window?

A. J-Type, 2 Fields: Opcode(6), Target Address(26)
B. Opcode = 0x2 (Looked up in Sheet)
C. Label = GREQ, Address = 0x00400030
D. Addressing Info
    a. 26 bits for address field
    b. Using how the PC works and each address is a mult. of 4, we can determine the 6 additional bits needed for a 32 bit address.
        i. First 4 bits of PC+1 = 0000
        ii. Since addresses multiple of 4, last two bits will be 00
        iii. 0000 <u>00000100000000000000001100</u> 00
        iv. 0000 0000 0100 0000 0000 0000 0011 0000
        v. 0x00400030
E. 0x0810000C <--> 0000 10<u>00 0001 0000 0000 0000 0000 1100</u>

(Assignment, individual) Conversion in proc2.s
Convert the following line in proc2.s to machine code and then back to MIPS instructions at the following addresses:
0x0040000c
0x00400014
0x0040002c
0x00400034