**CSE 140 Lab/HW#1 – By: Ilsia Gonzalez**

# 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

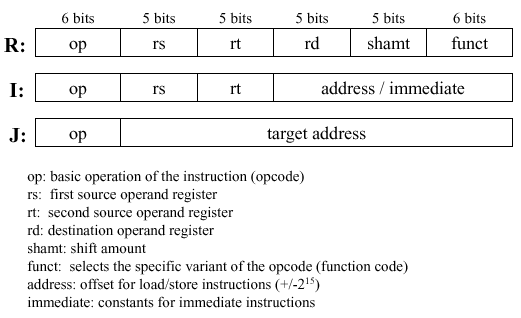
Find a classmate to work with and perform the following tasks (You are allowed to form a group of three):

1. Record the name(s) of your partner(s).
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?

* It has 32 bits.

1. 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.

* There are three different machine instructions which are R-format, I-format, and J-format.



* R-format:

This includes Arithmetic instructions. One example of R-format can be add $t0, $s1, $s2 which basically adds the two numbers stored in register 1 and 2. Then its stored in $t0. A second example of R-format can be sub $t1, $s1, $s2 which subtracts register 1 and 2. Then its stored in $t1.

* I-format:

This includes branch, transfer, and immediate instructions. One example can be lw $t0, 52($s3). This operation loads a word into register $t0 through collecting the data located in $s3 and the number 52 is the offset. Another example is sw $t0, 52($s3) and this instruction does the opposite of the other instruction.

* J-format:

This includes jump instructions. Jump instructions include j and jal. These are used to specify the target of the jump. One example can be j Loop. Another example can be j 1028.

1. Now, locate the instruction in line #14 of proc1.s. Let’s translate this instruction into machine code.
   1. What instruction type is this? How many fields does this type of instruction have? What are the names of these fields?

* It is a R type instruction. R type instructions have six fields, and they include the following: op, rs, rt, rd, shamt, and funct. OP is the operation code. Rs, rt, and rd are the register addresses. Shamt is the shift amount. Funct is the function code.
  1. Refer to the MIPS sheet, what is the value of the opcode of this instruction in Hex? What register is rs? What is the value of this register in Hex? What register is rt? What is the value of this register in Hex? What register is rd? What is the value of this register in Hex? What is the value of the funct field of this instruction in Hex?
* Set Less Than (slt) opcode is all 0s and in Hex it is 0. Register rs is $t0 and is 8 in Hex. Register rt is $s1 and is 17 in Hex. Register rd is $s0 and is 16 in Hex. I think the funct field is 32 in Hex.
  1. 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.
* Binary: 0000 0001 0001 0111 1011 0000 0010 0000
* Hex: 0x0117b020

1. Now, let’s convert a machine code to a MIPS instruction. Locate address 0x00400024 from the Text Segment window.
   1. What is the machine code at this address in Hex? Convert this code into binary.

* Hex: 0x34240000
* Binary: 0011 0100 0010 0100 0000 0000 0000 0000
  1. 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?
* Instruction Type: I -Format
* # fields: 4
* Field names: op, rt, rs, address/immediate
  1. According to the binary machine code, what is the value of each field in Hex?
* Op: d
* Rs: 4
* Rt: 1
* Immediate: 0
  1. 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?
* It is operation ori because its represented in the Text Segment window. Also, the op code binary is 001101 and that is ori. The rs is $a0. The rt is $at. The immediate is value 0.
  1. What is the final MIPS instruction? Is it the same as the Source column in the Text Segment window?
* ori $a0 $at 0
* yes

1. Now, let’s take a look at line #17 of proc1.s.
   1. What format is this instruction?

* This is I-Format.
  1. What are the values of opcode, rs, and rt of this instruction in hex?
* Opcode: 5
* Rs: 8
* Rt: 0
  1. 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.)
* Target Label: LESS
* Address: 0x00000001
  1. So, do we put this address as the value of the immediate field of the instruction? Why?
* Yes, I think we can because the basic of this address is bne $8, $0, 0x00000001. However, I am not 100% sure.
  1. How do we find the value of the immediate field? What is this value?
* The immediate consists of a total of 16 bits. So when divided by 4 it is the last 4 numbers of the Address.
* Address Immediate Hex: 0014
* Binary: 0000 0000 0001 0100
  1. 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?
* Binary: 0000 0000 0100 0000 0000 0000 0001 0100
* Hex: 0x0200000 0000 0001 0100
* No, it doesn’t match.

1. Finally, let’s convert the j instruction in line #20.
   1. What format is this instruction? How many fields are there in this format?

* Its in J-Format and it has two fields.
  1. What is the opcode of this instruction in hex?
* Opcode in Hex: 2
  1. What label and address does this instruction jump to?
* Label: GREQ
* Address: 0x00400030
  1. 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?
* Well, we can use a total of 26 bits as the first 6 bits of the J-Format is specifically for the opcode. So, we would squeeze this by approaching the address backwards. Then place 0 for the remaining variables to make a total of 8 hex variables for the address.
* Address field binary: 00 0100 0000 0000 0000 0001 1000
  1. 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?
* Binary: 0000 0000 0100 0000 0000 0000 0001 1000
* Hex: 0x00400018
* Yes

# (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

* Machine Code: 0000 0010 0010 0010 0000 0000 0000 1100
* We know that it is in R format.
* R-Format: OP RS RT RD shamt funct
* MC to Instruction: 000000 10001 00010 00000 00000 001100
* MIPS: addu $s1, $v0

0x00400014

* Machine Code: 0000 0000 0100 0000 0000 0000 0001 0100
* We know that it is in I format.
* I-Format: OP RS RT Immediate
* MC to Instruction: 000000 01000 00000 00000 00000 010100
* MIPS: beq $t0, $zero, [ ] 🡨 This is LEEQ

0x0040002c

* Machine Code: 0000 0000 0100 0000 0000 0000 0010 1100
* We know that it is in J format.
* J-Format: OP Target Address
* MC to Instruction: 000000 00010000000000000000101100
* MIPS: j END

0x00400034

* Machine Code: 0000 0000 0100 0000 0000 0000 0011 0100
* We know that it is in I format.
* I-Format: OP RS RT Address/Immediate
* MC to Instruction: 000000 00010 00000 0000000000110100
* MIPS: lui $v0, $zero

You must show all the steps including values of the instruction fields in order to receive points.

Verify your answers with the Text Segment window.

# What to submit by when

When you are done with this lab assignments, you are ready to submit your work to CatCourse. Make sure you have included the following before you press Submit:

* Your answers for the exercise and the assignment in a pdf or MS Word document.
* Deadline: **11:59PM of one day before the next lab** (If this lab is assigned on 2/2, the deadline is 2/8 11:59PM)