**CS401-1 MIPS 2 – MIPS Assembler**

NAME: Battista, Jude

|  |  |  |
| --- | --- | --- |
| **CATEGORY** | **POINTS** |  |
| **Exercise 1: Assembling MIPS (Individual)** |  | 10 |
| **Exercise 2: Little/Big Endian (Individual)** |  | 10 |
| **Exercise 3: Disassembling MIPS (Individual)** |  | 20 |
| **Exercise 4: MIPS Test Program (Group)** |  | 30 |
| **Code Review / Walkthrough (Group)** |  | 30 |
| **TOTAL** |  | 100 |

## For this laboratory each of you individually will:

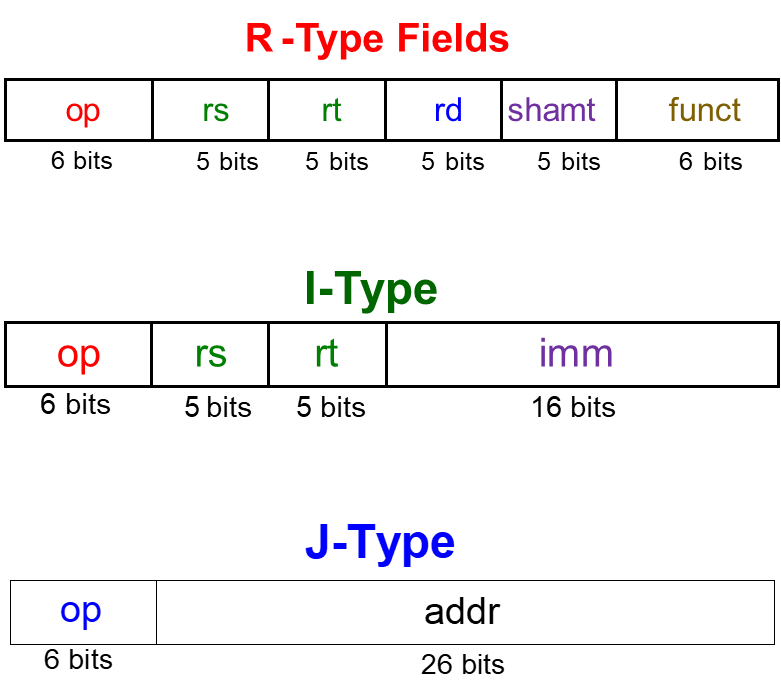
* Fill in the answers to the questions in this document and submit it on blackboard for assignment MIPS 2. Each must submit a copy of their own and do their own work for the first 3 exercises.
* Learn how Big/Little Endian works.
* **Learn how to assemble/disassemble MIPS programs by hand.**
* **Write a MIPS program that you will eventually re-write when you test your own processor.**

## Group project tips:

* Decide on a means of communication. Groups are setup on blackboard and it has tools for communication. Or you could use slack. In any case, make sure you have an efficient communication channel in case something happens.
* Decide on time(s) to meet. You will have time in class today, but not next class period. You will need to work on this outside of class.
* Google “pair programming kindergarten”. Read the PDF by Laurie Williams. Follow these guidelines and treat each other with respect. Listen to one another. If there are issues that can’t be resolved through the tips in this document please talk to your instructor.

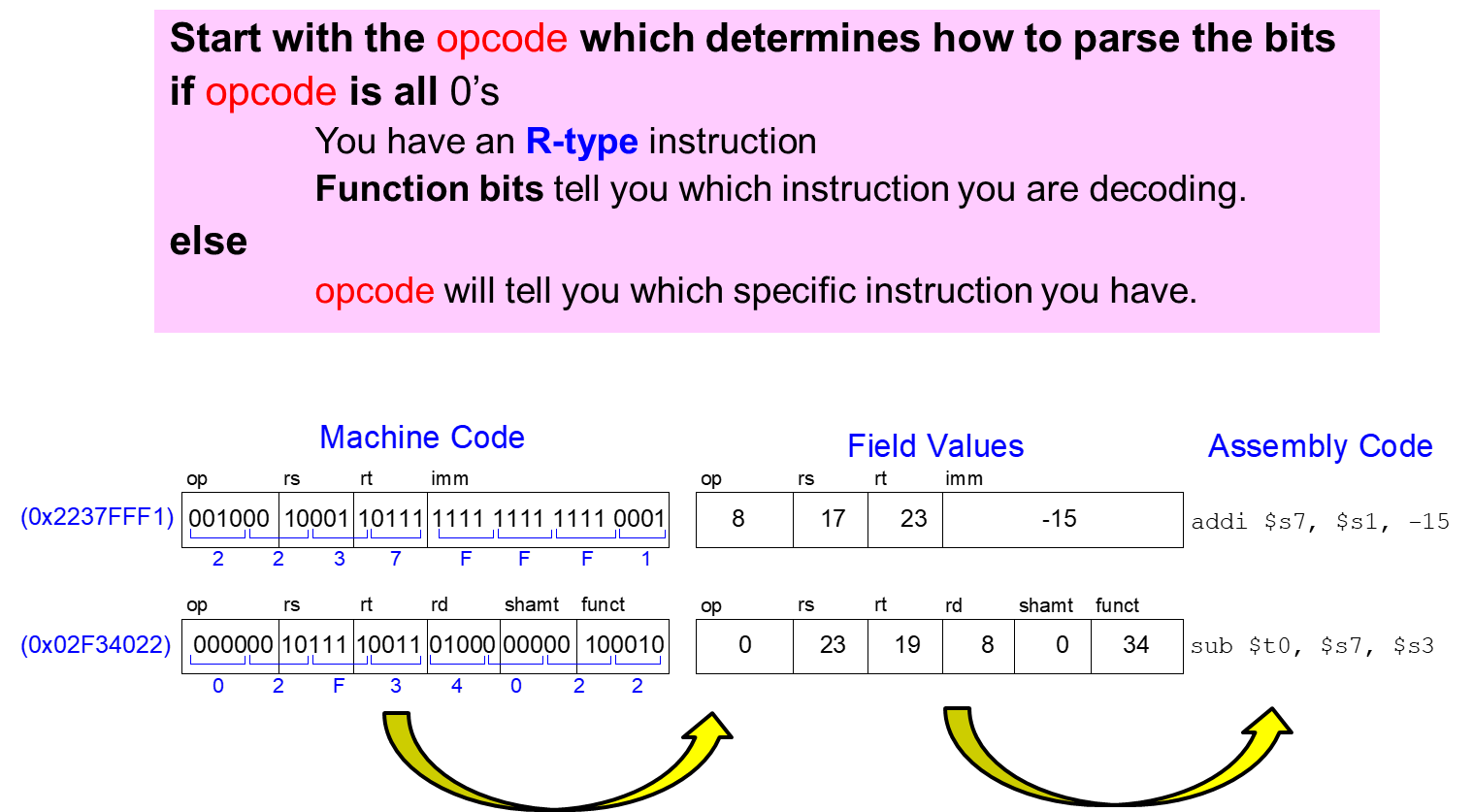
## Important: Resources for Converting MIPS Assembler to Machine Code

In order to convert from MIPS assembler to machine code manually, you will want to use Appendix B of your textbook. This Appendix has examples of the different MIPS instructions that are available. You will also want to refer to your textbook, the lecture notes and the layouts of the different types of instructions given here:



#### Exercise 1: Assembling MIPS Assembler by Hand (Individual)

To assemble MIPS assembler by hand, follow this procedure:



Given the following MIPS assembler program, translate each of the given instructions to MIPS machine code. The first one is done for you as an example. Enter your values directly in the tables below.

addi **$**s0, **$**0, 22 # $s0 = 0 + 22 = 22

addi **$**s1, **$**s0, 20 # $s1 = $s0 + 20 = 42

sll **$**s1, **$**s1, 2 # $s1 = $s1 << 2

sw **$**s1, 0x10**($**t2**)** # store contents of $s1 to (0x10+$t2)

addi **$**s0, **$**0, 22 # $s0 = 0 + 22 = 22

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field Values | (8) | 0 | 16 | 22 |
| Machine Code | 001000 | 00000 | 10000 | 0000 0000 0001 0110 |
| Size of Field | 6 bits | 5 bits | 5 bits | 16 bits |

addi **$**s1, **$**s0, 20 # $s1 = $s0 + 20 = 42

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field Values | 8 | 16 | 17 | 20 |
| Machine Code | 001000 | 10000 | 10001 | 10100 |
| Size of Field | 6 bits | 5 bits | 5 bits | 16 bits |

sll **$**s1, **$**s1, 2 # $s1 = $s1 << 2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field Values | 0 | 0 | 17 | 17 | 2 | 0 |
| Machine Code | 000000 | 00000 | 10001 | 1001 | 00010 | 000000 |
| Size of Field | 6 bits | 5 bits | 5 bits | 5 bits | 5 bits | 6 bits |

sw **$**s1, 0x10**($**t2**)** # store contents of $s1 to (0x10+$t2)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field Values | 43 | 10 | 17 | 0x10 |
| Machine Code | 101011 | 01010 | 10001 | 0000 0000 0001 0000 |
| Size of Field | 6 bits | 5 bits | 5 bits | 16 bits |

#### Exercise 2: Little Endian vs Big Endian Memory (Individual)

**Little endian machine**: Store the integer 0xA1B2C3D4 in the following memory starting at memory location 0x1000102C. Memory locations increase from bottom to top and from left to right...

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| BYTE ADDRESS | Byte +0 | Byte +1 | Byte +2 | Byte +3 |
| 0x10001034 |  |  |  |  |
| 0x10001030 |  |  |  |  |
| 0x1000102C | D4 | C3 | B2 | A1 |

**Little endian machine**: Store the null terminated string “Endian” in the following memory starting at memory location 0x1000102C. Memory locations increase from bottom to top and from left to right...

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| BYTE ADDRESS | Byte +0 | Byte +1 | Byte +2 | Byte +3 |
| 0x10001034 |  |  |  |  |
| 0x10001030 | 61 | 6E | 00 |  |
| 0x1000102C | 45 | 6E | 64 | 69 |

**Big endian machine**: Store the integer 0xA1B2C3D4 in the following memory starting at memory location 0x1000102C. Memory locations increase from bottom to top and from left to right...

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| BYTE ADDRESS | Byte +0 | Byte +1 | Byte +2 | Byte +3 |
| 0x10001034 |  |  |  |  |
| 0x10001030 |  |  |  |  |
| 0x1000102C | A1 | B2 | C3 | D4 |

**Big endian machine**: Store the null terminated string “Endian” in the following memory starting at memory location 0x1000102C. Memory locations increase from bottom to top and from left to right...

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| BYTE ADDRESS | Byte +0 | Byte +1 | Byte +2 | Byte +3 |
| 0x10001034 |  |  |  |  |
| 0x10001030 | 61 | 6E | 00 |  |
| 0x1000102C | 45 | 6E | 64 | 69 |

#### Exercise 3: Disassembling Mips Machine Code (Individual)

Consider the following program. Convert it from MIPS machine code to assembler using Appendix B of your textbook.

ADDRESS MACHINE CODE

[0040002c] 20100000 addi $s0, $0, 0

[00400030] 2011000a addi $s1, $0, 10  
[00400034] 02118020 add $s0, $s0, $s1  
[00400038] 2231ffff   addi $s1, $s1, 65535 //0xffff  
[0040003c] 1e20fffe   bgtz $s1, 0xfffe

Write the MIPS Assembly language for your program here:

addi $s0, $0, 0

addi $s1, $0, 10

add $s0, $s0, $s1

addi $s1, $s1, 65535

bgtz $s1, 0xfffe

Write the code for a C++ program that does exactly the same thing here:

int s0 = 0 + 0;

int s1 = 0 + 10;

do {

s0 += s1;

s1--;

} while (s1 > 0)

#### Exercise 4: MIPS Assembler Test Program Design

Think about the types of applications that you hope your processor will be able to execute. You have been given the task of writing a test program that will help verify your processor is working correctly. Think about the types of instructions you might want to use. What instructions does MIPS have that are similar to the ones that you will need to use?

Negotiate in your group of two as to the type of MIPS test program you will want to develop. Come up with an interesting algorithm on your own. **Do NOT simply copy a MIPS program from the internet while it may be tempting, doing this will result in an immediate 0 for this entire assignment**. The idea here is for you to have fun and create something entirely original. This obviously must use MIPS assembly language and be able to run on QTSpim.

In the future you might want to convert this algorithm to your own assembly language, so use this time wisely and write something fun and interesting.

#### Exercise 5: Code Walkthrough ALU Presentations

On the due date for this design, groups of two will take 5 minutes (max) and present their test algorithm design to the class. You will be graded on whether you present the following items. **Do not use more than 4 or 5 slides to summarize your algorithm.**

Minute 1: What MIPS instructions you chose for your algorithm and why you chose them.

Minute 2: A neat assembly language listing of your program in the MIPS language. You must include comments.

Minute 3: A demonstration of your MIPS assembler language running.

Minutes 4 - 5: An analysis of your algorithm.

* What did you like / dislike about the MIPS instructions you used?
* How efficient was your algorithm?
* Do you think you might use this algorithm for your final project demonstration or will something like it be used for testing?