# CS 401 NAME(s): Due Apr 17

## FINAL GROUP PROJECT: CPU DESIGN PART I: LANGUAGE DESIGN

|  |  |  |  |
| --- | --- | --- | --- |
| **CATEGORY** | **Beginning**  **70% – 79%** | **Satisfactory**  **80% – 89%** | **Excellent**  **90% – 100%** |
| 30 pts possible  **1.1 PROCESSORS OF THE WORLD COMPARISON** | Compared 2 processors and/or poor quality comparison of processors. | Compared 4 processors architecture, machine code, and assembly language with adequate comparison. | Neat, Compared 4 processors architecture, machine code, and assembly language with an excellent comparison and discussion of pros and cons of each processor. |
| 30 pts possible  **1.2 PROGRAMMERS VIEW OF THE ARCHITECTURE** | Registers and 5-7 non-MIPS CPU instructions. **OR**  11-12 MIPS instructions | Registers and 8-9 non-MIPS CPU instructions with syntax table and description **OR**  13-14 MIPS instructions with syntax table and description | Registers and 10 or more non-MIPS CPU instructions with syntax table and description **OR**  15 or more MIPS instructions with syntax table and description |
| 30 pts possible  **1.3 TWO ASSEMBLY LANGUAGE TEST PROGRAMS** | 1 ad-hoc program that tests all instructions | 1 ad-hoc program that tests all instructions AND  1 adequate well-known algorithm | 1 complete ad-hoc program that tests all instructions with comments AND  1 complete well-known algorithm with comments |
| 10 pts possible  **1.4 HEXADECIMAL MACHINE CODE LISTINGS** | 1 hex listing | 2 hex listings | 2 hex listings |
| **10 bonus pts possible**  ASSEMBLER PROGRAM | No assembler | Adequate assembler with comments | Excellent assembler with comments |

For this project you must work in a group of three to four individuals.

1.1 Processors of the World Comparison

Compare the hardware architectures and assembly/machine languages for four different processors. You must include ARM and MIPS in your comparison however, you may pick any two other processors you wish; below are some suggestions with info:

1. Old Style Intel 8080 or 8085 Architecture
   * 1. <http://en.wikipedia.org/wiki/Intel_8080>
     2. <http://www.intel-vintage.info/intelotherresources.htm#906748189>
2. Old Style Motorola 6502 Architecture
   * 1. <http://www.visual6502.org/welcome.html>
     2. <http://en.wikipedia.org/wiki/MOS_Technology_6502>
     3. <http://opencores.org/project,t65>
3. Modern General Purpose MIPS Architecture – see your textbook.
4. Modern General Purpose ARM Architecture
   * 1. <http://www.apdl.co.uk/riscworld/volume1/issue4/armcode/index.htm>
     2. <http://en.wikipedia.org/wiki/ARM_architecture>
     3. <http://morrow.ece.wisc.edu/ECE353/arm_reference/ddi0100e_arm_arm.pdf>
5. Modern Application Specific Nvidia Fermi Architecture (Graphics Processor)
   * 1. <http://www.orangeowlsolutions.com/archives/388>
     2. <http://www.nvidia.com/content/PDF/fermi_white_papers/NVIDIA_Fermi_Compute_Architecture_Whitepaper.pdf>
     3. <http://code.google.com/p/asfermi/wiki/Opcode>
     4. <http://code.google.com/p/asfermi/wiki/nanb>

When comparing the four architectures you should look at the high-level architecture design of each style and compare the following:

1. Data Path Units – Write a paragraph describing the Data Path Unit architecture for each processor. How do the DPUs compare one to another?
2. Instruction Format Comparisons – Pick two instructions: Pick a branch instruction and one other instruction. Compare the assembly and machine code format for both of these instructions among all 4 processors.

1.2 CPU Design: Assembly Language Design and Format

Step 1: Decide on the specific application area you want your processor to focus on. Are you interested in general purpose? audio? graphics? encryption? security? Look at sample processors and their assembly language design for ideas on how you want to design your assembly language. If you choose MIPS, your application area is general purpose.

Step 2: Design the programmer’s view of the architecture for your processor (e.g. the general instructions and instructions for accessing registers/memory). If you choose MIPs, this is done for you, however, you will need to implement a minimum of 5 additional instructions.

Don’t start with too many instructions! Look on pages 432 and 433 in the text book to see the instructions implemented in the book for the single cycle processor, (e.g. addi, or, and, add, beq, slt, sub, sw, lw, j). Your processor will need enough total instructions to do useful work (somewhere in the range of 5 to 10 instructions)

Once you have a list of instructions (and registers) for your assembly language, write some ad-hoc assembly language programs (using your newly defined assembly language) that you will use to test your CPU once you have designed and implemented a processor to run your language. Obviously, you MUST understand how your language will work BEFORE you implement a CPU to decode and run your language!

Assembly Language Common Requirements:

* You must have enough instructions to run a useful program (e.g. something of the complexity of generating the Fibonacci sequence)
* If you decided to simply implement a MIPS style processor (i.e. expand the single cycle example), then you must add an additional 5 instructions chosen from Appendix B over and above the instructions you have already implemented.

Two Assembly Language Test Programs:

1. Implement an ad-hoc assembly program that runs every instruction you have designed.
2. Implement a well know algorithm that you will eventually run on your CPU (e.g. sieve of Eratosthenes, Fibonacci etc.)Implement a version of this algorithm in your assembly language.

Step 3: Translate your program from assembly language into machine language so that it will be ready to run once you have implemented your microprocessor design in VHDL. *For bonus points, make an assembler that will translate any assembly language program using your instructions into your machine code.*

### What to Hand In:

* Place the following items in a text document (i.e. word / pdf) called FP1\_Language\_Design in your folder on CS1. Hand in a single printout of the front page of this document with your names on the document.
* The processors of the world comparison of 4 processors.
* The programmers view of the architectures: define the registers, and the binary format for each instruction. (e.g. fig 6.10, 6.12, etc) and a table of instructions that describes each instruction’s operation (see appendix B for an example of what this should look like).
* Two Assembly Language Test Programs, an ad-hoc program and a well-known algorithm. For full points these must be complete and well documented.
* A machine code listing (in Hexadecimal) of each of your programs.
* For Bonus Points: Your assembler code (if implemented)