EEC 170

Assignment 2 – Chapter 2

1. **(6 pts)** Each of the following machine codes are instructions in a RISC-V processor implementing the RV64I ISA described in Chapter 2 of Patterson & Hennessey. For every instruction provide the following information:
   1. The instruction name and it’s format.
   2. Assign a hexadecimal value to each instruction field.
   3. Describe what the function does.
      1. Explain the effect the instruction has on the CPU’s registers, memory, and/or the next instruction to be executed. If there is no effect on any of these components, explain why.
   4. **Hint:** Translate the instruction into binary first.
   5. **Hint:** Begin, as the CPU does, with the opcode field, which reveals an instruction’s format.

Instruction 1 **(3 pts)**: **0x0FF00093**



Type: Immediate type; addi

Immediate: 0xFF = 255, Rs1: x0, Rd1: x1

Instruction: **addi x1, x0, 255 # x1 = x0 + 255 = 255**

CPU registers: Changes the contents of register x1 to 255.

Memory: Not a memory operation.

Next instruction: Next instruction to be executed will be PC + 4.

Instruction 2 **(3 pts)**: **0x00000113**



Type: Immediate type; addi

Immediate: 0, Rs1: x0, Rd1: x2

Instruction: **addi x2, x0, 0 # x1 = x0 + 0 = 0**

CPU registers: Changes the contents of register x2 to 0.

Memory: Not a memory operation.

Next instruction: Next instruction to be executed will be at PC + 4.

1. **(5 pts)** A common practice in assembly programming is to extract segments of bits within a word. For example, in the HiFive Unleashed RISC-V development board, the UART receiver peripheral (a component outside of the CPU) stores received 8-bit character data in a word-sized memory block as shown below:

Table

Description automatically generated

* 1. **(2 pt)** If all that was needed from this word was the character data (the last 8 bits), what single instruction could be used to retrieve it? Why?  
       
     Load byte (**lbu**) is most appropriate because load word would require an additional instruction to mask bit 31.

* 1. **(3 pts)** Unfortunately, the above method won’t work because the final bit of the word is needed to determine if the received character is valid. Assume the word is located at memory location 0x10010004. Write a short RISC-V program that performs the following steps:  
     1. Load the word from memory into register x5.
     2. Using only RISC-V logical operations, extract the ASCII “Received data” byte into register x6 and the “Receive FIFO empty bit” into register x7.
     3. **Note:** If bit 31 is 1, x7 should have a magnitude of 0x0000\_0001, not 0x1000\_0000.
     4. **Note: this is the first time in the semester your points may be reduced for not following the style guide! Going forward, make sure all code is properly formatted. Refer to the style guide!**



Note that the solution is the last three lines and the first two instructions to create the address. The first two instructions may be converted to a pseudo instruction.

1. **(2 pts)** Stacks grow from high addresses to low addresses. Adding to the stack pointer pops an element in the stack. Heaps grow upwards in memory from the end of the static data memory segment.
2. **(5 pts)** Translate the following C-code into a set of RISC-V instructions. Annotate the purpose of each instruction with respect to the C-code as a comment.





1. **(2 pt)** The previous program runs on a CPU in which each branch instruction carries a CPI of 6, each arithmetic instruction carries a CPI of 1, load/store instructions have a CPI of 10, and all other instructions carry a CPI of 4.
   1. If the CPU has a clock rate of 500 MHz, what is the execution time of the program?   
        
      For the solution above, if the entire program’s execution is considered, there will be 2 addi followed by 5 iterations of 2 branch and 1 addi. On the final iteration, a single branch instruction will execute to break from the loop. Therefore, a total of 7 arithmetic and 11 branch instructions will occur. Points were also awarded if analysis was only performed on a single loop iteration due to an error in the original solution.
   2. Use Amdahl’s law to show whether it is possible to reduce the CPI of branch instructions such that execution time is reduced by 10% or more.

CPI relates to ET improvement like so:

Yes, because 1.124 is greater than 0.