COMP 3370 Assignment 1

Trevor Bekolay, 6796723

Simulator Design

The CPU that is reflected in my code is laid out rather simply, utilizing the common features of almost all CPU's to fulfill the goal of running the ISA provided in the original specifications. It contains the following registers:

- 16 general purpose registers
  - → Used to temporarily store values
- Program counter register
  - → Stores the location of the current instruction, or the next instruction to be fetched, depending on the state. In general, it keeps track of where we are in the program.
- Instruction register
  - → Stores the next instruction to be decoded, or the current instruction, depending on the state.
- Branch register
  - → 2 bit register containing information about branches (eg, was branch taken) to determine the PC's value
- Store register
  - → 1 bit register that determines if a value is stored in memory or register
- Output register
  - → 16 bit register that holds output produced by the execution step

The simulator begins with the program counter at 0x0000, and goes through a sequence of four steps repeatedly until an error is generated, and the CPU halts operation. Throughout these steps, the data follows a set path that can be referred to as the "datapath." The steps are as follows:

- Fetch
  - → The location in main code memory that is pointed to by the program counter register is copied into the instruction register.
- Decode
  - → The value in the instruction register is broken up in accordance with the ISA.
- Execute
  - → Parts of the instruction are sent to other sections of the CPU, such as the ALU.
  - → If the instruction is a branch, the branch register will be set.
  - → If the instruction is a move to memory, the store register is set.
  - → The program counter is incremented
- Writeback
  - → The output from the ALU or the control unit (eg, if the output comes directly from a register or the instruction) is stores in some kind of memory.
    - This may be the main data memory, a register or the program counter.
    - It depends on the state of the store and branch register where the output is stored.

After this process is complete, it repeats, normally beginning with fetching the next instruction in the code space because of the incrementation of the program counter. If the last operation was a branch or jump, then execution will continue where the program counter now points.

Whenever an error occurs, whether it be an illegal opcode, out or range memory access, or anything else, a piece of memory will be written that contains the program's exit code, detailing the reason that it failed. The value in the program counter register is available to be scrutinized for debugging purposes.

Finite State Machine