**Instruction Set and Instruction Encoding:**  
The designed 4 bit microprocessor uses memory with 8 bit wide rows, therefore each individual instruction is 8 bit wide. The implemented instructions are as follows:

Load immediate value to register   
Jump (unconditional)  
Jump (if ALU operation produces carry)  
Addition (Ra+Rb)  
Subtraction (Ra-Rb)  
No operation (Implemented by putting the register selection decoder into a state where no register is connected)  
  
The 8 bit instructions are encoded as follows:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **J** | **C** | **D1** | **D0** | **Sreg** | **S/I3** | **I2** | **I1** |

**J:** The seventh bit of the instruction is for jump, whenever the J bit it set, the custom inputs from the EEPROM are passed onto the program counter and loaded, hence implementing a jump instruction   
  
**C:** The sixth bit is fixed for the “Jump if carry” operation, this bit being set will load the custom input from the EEPROM into the program counter if the ALU produces a carry. This is implemented via a combinational logic circuit connected at the input pin of the binary counter which acts as a program counter in the design.

**D1D0:** The fifth and fourth bit of the instruction are to select the register address, 00 corresponds to Ra, 01 corresponds to Rb, 10 to Rout and 11 corresponds to no register. Therefore, setting D1D0 to 11 is effectively a “no operation” instruction.

**Sreg:** The third bit is to multiplex the register inputs between the output of the ALU and the custom inputs from the instruction. 0 selects the ALU output to be passed onto the registers and 1 selects the custom input to be passed onto the registers.  
  
**I3I2I1:** The first three bits of the instruction correspond to the custom input, this may either be an immediate value to be loaded into the registers or the address of the instruction to be jumped to. Due to space limitations in the instruction width, the maximum value of an immediate that can be loaded is 7. Likewise, the maximum address that can be jumped to is also 7.

**S:** This bit decides whether the ALU performs addition or subtraction, since during ALU operation, the custom input is irrelevant and vice versa, this bit may be chosen for this dual purpose.

**Programs:  
  
Upcount:  
Psuedo code:**0 Ra = 0  
1 Rb = 12 Rout = Ra  
3 Ra = Ra+Rb  
4 jump to 2  
  
**Machine Code:**00001000 08  
00011001 19  
00100000 20  
00000000 00  
10110010 b2  
  
**Fibonacci Numbers:  
Psuedo code:**0 Ra=0 // load 0 to Ra  
1 Rb=1 // load 1 to RB  
2 Rout=Ra // Ra to output   
3 Rb=Ra+Rb // add Ra and Rb   
4 If carry jump to 000 //Restart if carry (overflow)   
5 Ra = Ra+Rb //swap Ra and Rb  
6 Rb = Ra-Rb //swap Ra and Rb  
7 Ra = Ra-Rb //swap Ra and Rb  
8 Jump to 2 //Jump to instruction 2  
  
**Machine Code:**

00001000 08  
00011001 19  
00100000 20  
00010000 10  
01110000 70   
00000000 00  
00010100 14  
00000100 04  
10110010 B2