**Part A**

1. Name of architecture:

Overall philosophy:

Specific Goals strived for and achieved:

|  |  |  |  |
| --- | --- | --- | --- |
| Instruction | OP code | Format | Example |
| lw | 000 01 x y | lw Rx, Ry | Rx = M[Ry]  Rx = $1, $2; Ry = $1, $3 |
| bgtR1 | 000 1 x ii | bgtR1 Rx, imm | Pc = pc + imm if Rx > $1  Rx = $2, $3; imm = [1:4] |
| halt | 000 0000 | halt | Pause |
| sw | 001 x iii | sw Rx, imm | M[imm] = Rx  Rx = $1, $2; imm = [0:7] |
| sub | 010 0 xx y | sub Rx, Ry | Rx = Rx + Ry  Rx = [$1-$4]; Ry = [$1, $3] |
| shiftL | 010 10 xx | shiftL Rx | Rx << 1  Rx = [$1-$4] |
| shiftR | 010 11 xx | shiftR Rx | Rx >> 1  Rx = [$1-$4] |
| init | 011 xx ii | init Rx, imm | Rx = imm  Rx = [$1-$4]; imm = [0:3] |
| bltR1 | 100 1 x ii | bltR1 Rx, imm | Pc = pc + imm if Rx < $1  Rx = $2, $3; imm = [1:4] |
| add | 100 0 xx y | add Rx, Ry | Rx = Rx + Ry  Rx = [$1,$2]; Ry = [$1-$4] |
| addi | 101 xx ii | addi Rx, imm | Rx = Rx + imm  Rx = [$1-$4]; imm = [-2:1] |
| j | 11 iiiii | J Label | PC = imm  Imm = [0:31] |

1. Registers design: 4 registers $1, $2, $3, $4.

$0: where is register 0?

1. Control flow:
2. Data memory addressing modes.

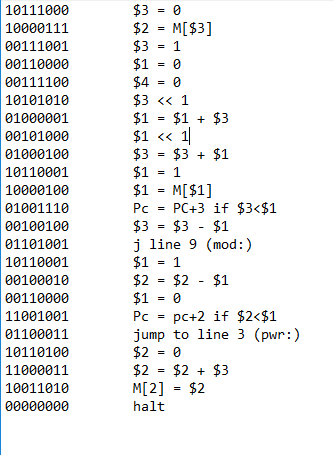
**Part B**

1. Compare:

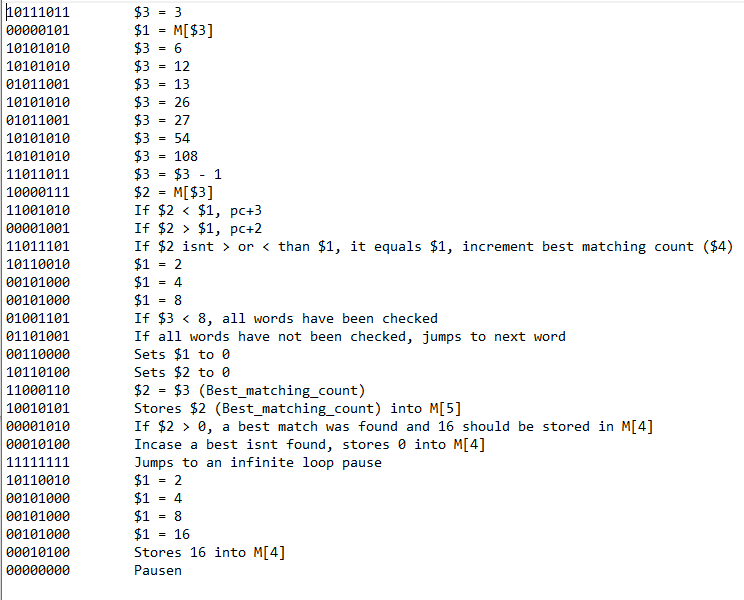
**Part C:**

1. Algorithms
2. Machine code for programs
3. Output of python disassembler

Program 1:



Program 2:



4.

