In this program, we will simulate the execution of the program by calculating the multiplexor settings for each instruction. Since the single-cycle processor executes one instruction at a time, there are no data hazards to consider. Here is a diagram of the processor showing the control signals needed to execute instructions:



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| A | Jump | 1 = this is a j instruction; 0 otherwise | G | Mem Write | | 1 = store instruction |
| B | Reg Write | 1 = this instruction writes back to the register file | H | WB Sel | | 0 = write back from ALU; 1 = write back from memory |
| C | ALU Src | 0 = Register read data B, 1 = sign extender | I | Rd/Rt Sel | | 0 = R-type instruction; 1 = not R-type |
| D | Forward A | not in single cycle | J | BEQ Branch | | 1 = beq instruction |
| E | Forward B | not in single cycle | K | ALU Control | | 0 = add; 1 = subtract; 2 = << 16 immediate; 3 = or immediate |
| F | Mem Read | 1 = load instruction |  |  |  | |

Program:

|  |  |  |  |
| --- | --- | --- | --- |
| **Address** | **Instruction** | **Mnemonic** | **Arguments** |
| 0x00400000 | 0x3c011001 | lui | $at, 0x1001 |
| 0x00400004 | 0x34300000 | ori | $s0, $at, 0x0000 |
| 0x00400008 | 0x8e080000 | lw | $t0, 0($s0) |
| 0x0040000c | 0x20090003 | addi | $t1, $zero, 3 |
| 0x00400010 | 0x11200004 | beq | $t1, $zero, 0x0004 |
| 0x00400014 | 0x01094020 | add | $t0, $t0, $t1 |
| 0x00400018 | 0x2129ffff | addi | $t1, $t1, -1 |
| 0x0040001c | 0xae080000 | sw | $t0, 0($s0) |
| 0x00400020 | 0x08100004 | j | 0x00400010 |
| 0x00400024 | 0x2002000a | addi | $v0, $zero, 10 |
| 0x00400028 | 0x0000000c | syscall |  |

Here is a list of instructions and what the control signals should be for these lines

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instruction | Op | Fct | Rd | Rs | Rt | Jump (A) | Reg Write (B) | ALU Src (C) | Mem Rd (F) | Mem Wr (G) | WB Sel (H) | Rd/Rt Sel (I) | Beq (J) | ALU (K) |
| add | 000000 | 100000 | 2 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| addi | 001000 | - | 0 | 1 | 16 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| beq | 000100 | - | 0 | 16 | 8 | 0 | 0 | 0 | 0 | 0 | - | 1 | 1 | 1 |
| j | 000010 | - | 0 | 0 | 9 | 1 | 0 | - | 0 | 0 | - | - | 0 | - |
| lui | 001111 | - | 0 | 9 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | - |
| lw | 100011 | - | 8 | 8 | 9 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| ori | 001101 | - | 31 | 9 | 9 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | - |
| sw | 101011 | - | 0 | 16 | 8 | 0 | 0 | 1 | 0 | 1 | - | - | 0 | 1 |
| syscall | 000000 | 001100 | 0 | 0 | 16 | 0 | 0 | - | 0 | 0 | - | - | 0 | - |

Your code should read the machinecode.txt file and reproduce the following table of control signal settings:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Machine Code | Instruction | Rd | Rs | Rt | Op | Fct | Jump | RegWrite | ALU Src | Mem Rd | Mem Wr | WB Sel | Rd/Rt Sel | beq | ALU |
| 0x00400000 | 0x3c011001 | Lui $at, 0x1001 | 2 | 0 | 1 | 0x0f |  | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0x00400004 | 0x34300000 | Ori $s0, $at, 0x0000 | 0 | 1 | 16 | 0x0d |  | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0x00400008 | 0x8e080000 | Lw $t0, 0($s0) | 0 | 16 | 8 | 0x23 |  | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0x0040000c | 0x20090003 | Addi $t1, $zero, 3 | 0 | 0 | 9 | 0x08 |  | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0x00400010 | 0x11200004 | Beq $t1, $zero, 0x0004 | 0 | 9 | 0 | 0x04 |  | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0x00400014 | 0x01094020 | Add $t0, $t0, $t1 | 8 | 8 | 9 | 0x00 | 0x20 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0x00400018 | 0x2129ffff | Addi $t1, $t1, -1 | 31 | 9 | 9 | 0x08 |  | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0x0040001c | 0xae080000 | Sw $t0, 0($s0) | 0 | 16 | 8 | 0x2b |  | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0x00400020 | 0x08100004 | J 0x00400010 | 0 | 0 | 16 | 0x02 |  | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x00400024 | 0x2002000a | Addi $v0, $zero, 10 | 0 | 0 | 2 | 0x08 |  | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0x00400028 | 0x0000000c | syscall | 0 | 0 | 0 | 0x00 | 0x0c | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |