CS 224
Section No.: 2
Spring 2018
Lab No.: 4
EFE ACER / 21602217

Preliminary Design Report

question 1:

- A **single-cycle** processor carries out an entire instruction in a single clock cycle. Hence, the clock cycle should be limited by the slowest instruction to avoid undesired behavior. Such processors do not require any non-architectural state.

- A **multi-cycle** processor allows instructions to take multiple clock cycles. These processors execute one instruction at a time, but the instruction is actually executed in a series of shorter cycles. Naturally, complicated instructions are executed in more cycles than the simpler ones. Multi-cycle microarchitecture is generally cheaper than a single-cycle architecture in terms of its hardware. The reason being that the multi-cycle architecture can reuse expensive hardware components while executing a single instruction. However, additional non-architectural registers are needed for these processors, to hold intermediate values.

- A **pipelined** processor applies so-called "pipelining" to a single-cycle microarchitecture. Therefore, it can execute multiple instructions in a parallel manner, which makes it significantly more efficient than the single-cycle processor. Additional logic is added to the hardware of such processors, so that dependencies between simultaneously executed instructions are safely handled. Several non-architectural registers are also needed for these processors. Having said all that, the cost of the additional hardware does not really matter since pipelined processors are essential for high-performance computing.

question 2:

- Main units that constitute the datapath:
  - **Program Counter Register (PC):**

    A register that contains the address of the instruction to be executed.

  - **Instruction Memory:**

    A memory component that reads the address of the instruction from the PC and fetches, outputs, the corresponding 32-bit instruction.

- **Register File**

  A component, where the general-purpose registers are located. It allows access to the registers specified in the instruction (rs, rd, rt, etc.). The contents of the accessed registers are later used as the operands of the instruction. It also allows modifying the contents of those registers, which is necessary for most instructions.

- **ALU**

  A component called Arithmetic Logic Unit, ALU for short; that involves combinatorial circuitry to perform arithmetic and logic operations on the operands specified in the instruction. It knows the operation to perform with the help of a control signal produced by the control unit and operands from the register file.

- **Data Memory**

  A memory component that stores the data being transferred to and from the rest of the data path.

- **Sign Extend**

  A component that involves circuitry to copy the most significant bit of the 16-bit immediate field of an instruction and put it into the upper 16-bits of the 32-bit output signal it produces.

- **Several flip-flops, multiplexers, adders and shift registers**

  These components are used in several places to handle dependencies between the other modules.

➢ Main units that constitute the control unit:

- **Main Decoder:**

  A decoder, which produces an output from the opcode bits of the instruction. It also provides a 2-bit ALUOp signal that is used later by the ALU Decoder to determine the correct operation to perform. It also provides information about the register and memory usage in the output bits it produces.

- **ALU Decoder:**

  This decoder uses the ALUOp signal generated by the Main Decoder, together with the funct bits of the instruction and produces an output such that the ALU knows the operation to perform.

question 3:

- Signatures of the modules:
    - PCRegister(input CLK, input PC’[32], output PC[32])
    - InstructionMemory(input PC[32], ouput Instr[32])
    - RegisterFile(input CLK, input Instr[25:21], input Instr[20:16], input WriteReg[5], output RD1[32], output RD2[32])
    - ALU(input SrcA[32], input SrcB[32], output ALUResult[32], output Zero)
    - DataMemory(input CLK, input ALUResult[32], input WriteData[32], output Result[32])
    - SignExtend(input Instr[15: 0], output SignImm[32])
    - ControlUnit(input Op[6], input Funct[6], output MemtoReg, output MemWrite, output Branch, output ALUControl[3], output ALUSrc, output RegDst, output RegWrite)

      Control Unit consists of two submodules:

        - MainDecoder(input Opcode[6], output MemtoReg, output MemWrite, output Branch, output ALUSrc, output RegDst, output RegWrite, output ALUOp[2])
        - ALUDecoder(input ALUOp[2], input Funct[6], output ALUControl[3])

question 4:

Location (hex)	Machine Instruction (hex)	Assembly Equivalent
00	20020005	addi $v0, $zero, 0x0005
04	2003000c	addi $v1, $zero, 0x000C
08	2067fff7	addi $a3, $v1, 0xFFF7
0c	00e22025	or $a0, $a3, $v0
10	00642824	and $a1, $v1, $a0
14	00a42820	add $a1, $a1, $a0
18	10a7000a	beq $a1, $a3, 0x000A
1c	0064202a	slt $a0, $v1, $a0

20	10800001	beq $a0, $zero, 0x0001
24	20050000	addi $a1, $zero, 0x0000
28	00e2202a	slt $a0, $a3, $v0
2c	00853820	add $a3, $a0, $a1
30	00e23822	sub $a3, $a3, $v0
34	ac670044	sw $a3, 0x0044, $v1
38	8c020050	lw $v0, 0x0050, $zero
3c	08000011	j 0x0000011
40	20020001	addi $v0, $zero, 0x0001
44	ac020054	sw $v0, 0x0054, $zero
48	08000012	j 0x00000012