University of KwaZulu-Natal
School of Engineering

Examinations: June 2014
ENEL4DP : Digital Processes

Duration: **2 Hours** Marks: 100

Examiners : Prof H. Xu (Internal)
: Dr. M. Ohanga(External)

Instructions : 1. Answer all questions.
2. This is not an open book exam and no notes may be used, either electronic or handwritten.
3. Questions (4.1), (4.2), (4.3), (4.4), (5.1), (5.2(a)) , (5.2(b)) and (6) must be answered on the attached answer sheet, which must be detached and handed in with your answer book. Ensure that you fill in your student number.
4. Questions 1, 2 and 3 should be answered in the answer book.

Question 1 VHDL [10marks]

Write a VHDL entity and an architecture for an-8-bit, parallel-to-serial converter as shown in Fig.1.1 below.

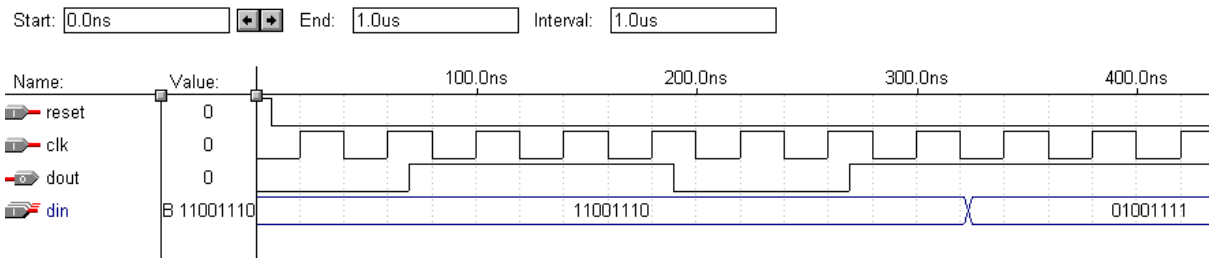


Fig. 1.1

Question 2 VHDL [10 marks]

Figure 2.1 shows part of the MIPS finite state machine from lecture. Control values not shown in each state are assumed to be 0. The entity of the finite state machine is given below. You are required to finish the architecture of the entity. The format of MIPS instruction is given as:

31-26:opcode	25-21: rs	20-16: rt	15-11: rd	10-6: shamt	5-0: func

Opcode 0 is R-type instruction; Opcode 4 is BEQ (Branch instruction).

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_SIGNED.ALL;
ENTITY CONTROL_FSM is
        Port (OP :in STD_LOGIC_VECTOR (5 downto 0);
            CLOCK, RESET: in STD_LOGIC; --clock reset
            PCWrite, IorD, MemRead, MemWrite: out STD_LOGIC;
            IRWrite, MemtoReg, PCSource : out STD_LOGIC;
            ALUOp, ALUSrcB : out STD_LOGIC_VECTOR(1 downto 0);
            ALUSrcA, RegWrite, RegDst: out STD_LOGIC);
END CONTROL_FSM;
```

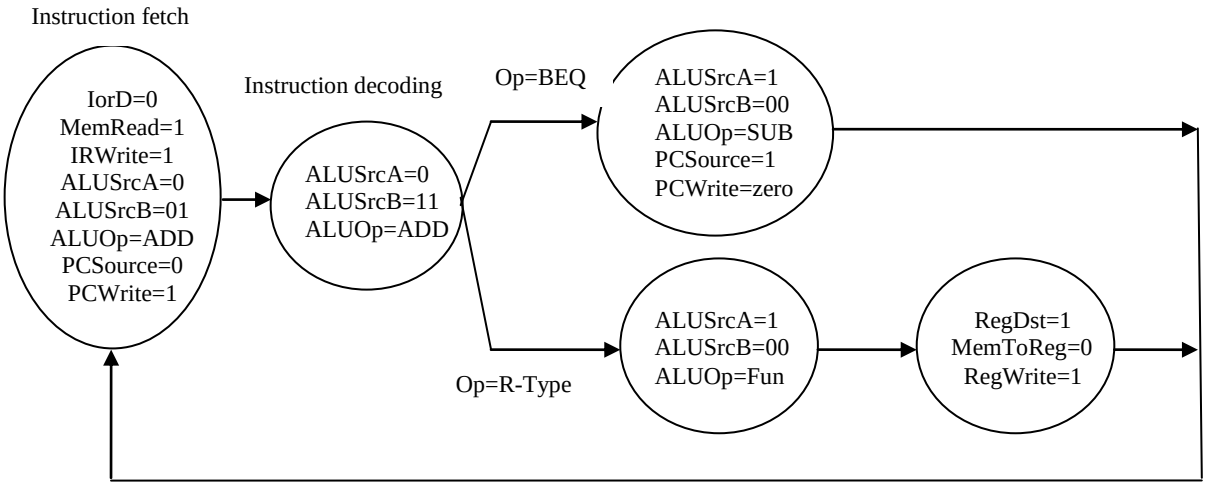


Figure 2.1 Part of the MIPS finite state machine

Question 3 VHDL [20 marks]

A register file is a collection of registers in which any register can be read or written by specifying the number of register in the file. There are two read ports and one write port. Suppose there are 4 registers in the register file. Each register contains 8 bits. An implementation of the read ports is shown in Fig. 3.1

(3.1) (**5 marks**) Write the full VHDL code to implement an 8-bit register;

(3.2) (**5 marks**) Write the full VHDL code to implement an 8-bit 4-to-1 multiplexer;

(3.3) (**10 marks**) Write the full VHDL code to implement the read ports by using the 8-bit register and 8-bit 4-to-1 multiplexer components.

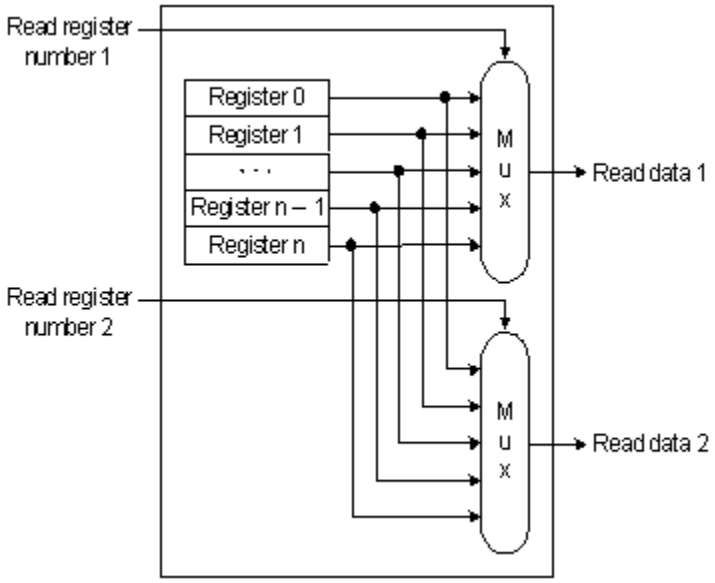


Fig.3.1 Implementation of the read ports

Question 4 Single Cycle Datapath Design (20 marks)

This question is based on the single-cycle datapath of MIPS shown in Figure 4.1.

(4.1) [**5 marks**] Clearly mark all wires that are active during the execution of **addi instruction**.

addi instruction format : addi $rt, $rs, imm # $rt <= $rs +imm

31-26:5 (decimal) : OP	25-21: rs	20-16:rt	15-0:imm

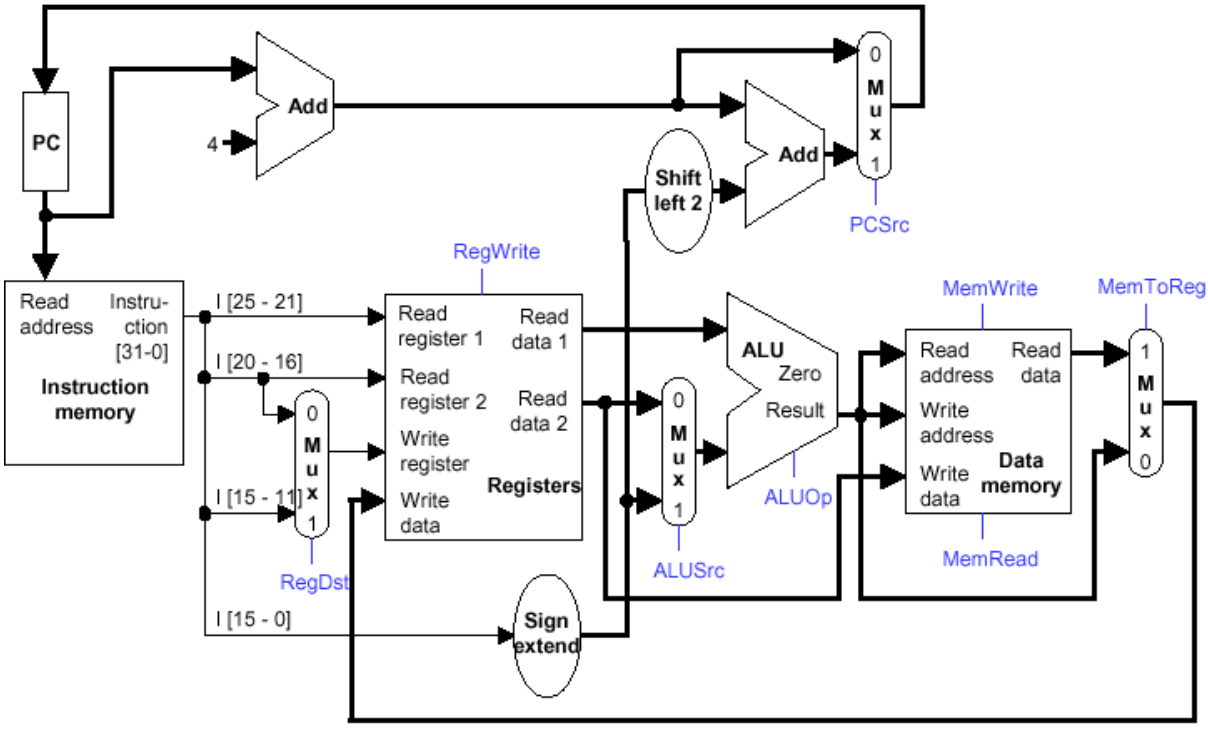


Figure 4.1 The single-cycle datapath of MIPS

(4.2) [**7 marks**] The single cycle datapath is shown in Figure 4.1. Show what changes are needed to support **jr instruction**. You should only add wires and multiplexers to the datapath; do not modify the main functional units themselves (the memory, Register file and ALU).

jr instruction format : *jr* $*rs* PC <= $*rs* ;$rt<=PC+4

31-26: OP=0	25-21: rs	20-16:rt	15-0:imm

(4.3)[**4 marks**] Specify how all control signals should be set for jr instruction. Use X for don't care. ALUOp can be ADD or SUB.

Instru.	RegDst	RegWrite	ALUSrc	PCSrc	MemRead	MemWrite	MemtoReg	ALUOp
jr								

(4.4)[**4 marks**] Given the functional unit latencies as shown in Table 4.1, calculate the minimum time in Table 4.2 to perform ***jr*** instruction.

Table 4.1

Function Unit	Latency
Memory read or write	2 ns
ALU	2 ns
Register File read or write	1 ns

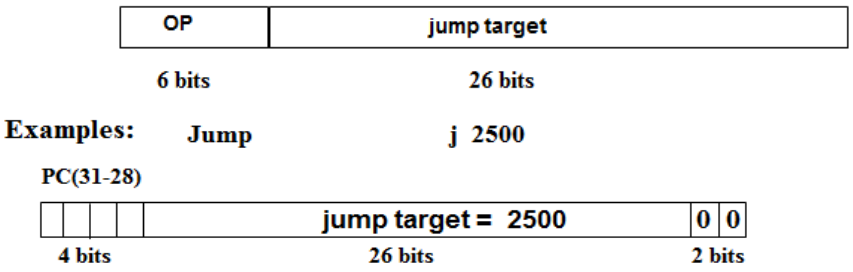
Table 4.2

Instruction	Minimum Time	Explain
jr		

Question 5 Multicycle Datapath [20 marks]

(5.1) [**5 marks**] **Jump** is executed in the multicycle datapath shown in Fig. 5.1. Assume that PC=(A000 0000)$_H$, and Memory(PC)=(0809 0000)$_H$, and Reg(n)=n. Complete Table 5.1 with 32-bit hex values, or **with an X for don't care** until the instruction completes. **This question is negative marking, 1mark/error.**

Jump-format instruction:



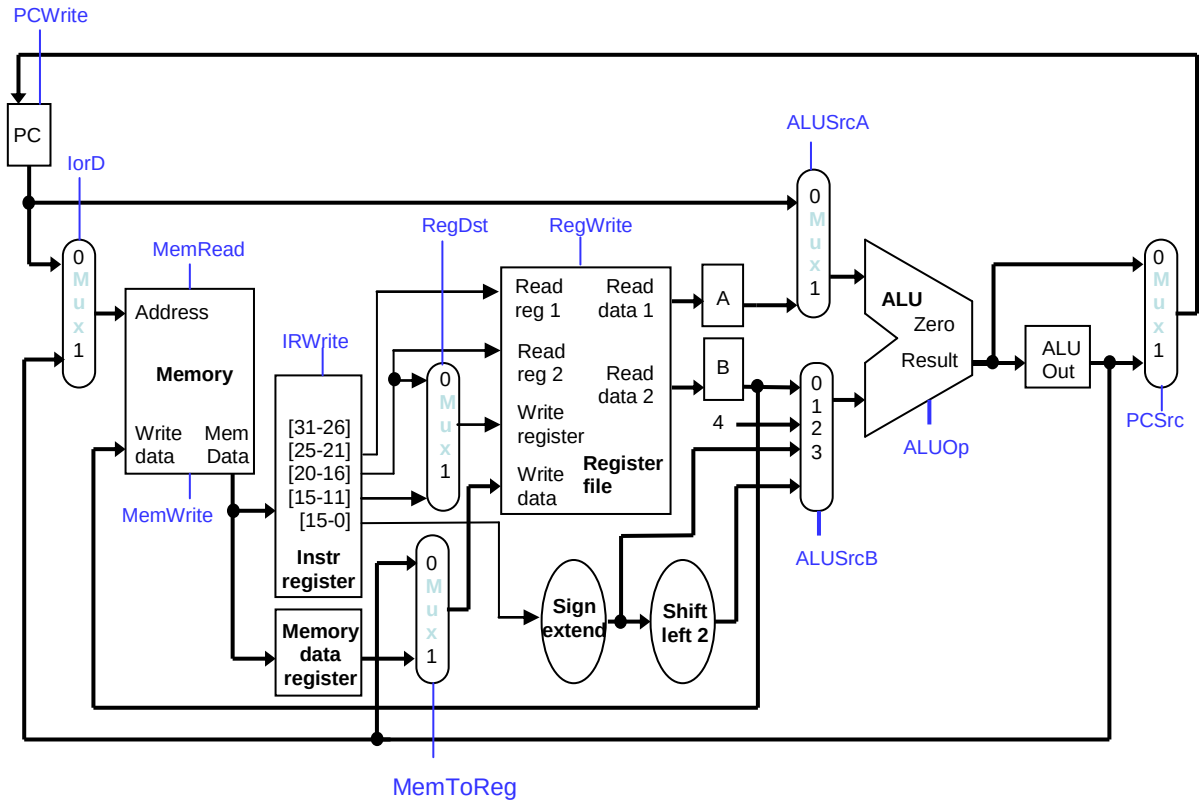


Fig. 5.1 The Multicycle datapath of MIPS

Table 5.1

Register	Clock 1	Clock 2	Clock 3	Clock 4	Clock 5
PC					
IR					
MDR					
A					
B					
ALUout					

(5.2) [15 **marks**] Assume that the ALU can perform the **max2** operation (i.e., return the greater of two inputs):

alu_result = A_input if A_input >B_input;

ALU operation for this instruction is **max2**.

Given this improved ALU, implement the **max4** instruction. The **max4** instruction writes the largest value of four registers into register **rd**:

max4 $rs, $rt, $rd, $rm $rd=max($rs, $rt, $rd, $rm)

Note that register rd is both an input and an output. Instruction **max4** has the following format:

31-26:op	25-21: rs	20-16: rt	15-11: rd	10-6: rm	5-0: func

(a) [7 marks] This question is based on the multicycle datapath of MIPS shown in Figure 5.1 . Show what changes are needed to support **max4**. You should only add wires and multiplexers to the datapath; do not modify the main functional units themselves (the memory, Register file and ALU).

(b) [8 marks] Complete the finite state machine diagram shown in Figure 5.2 for the **max4** instruction. Control values not shown in each state are assumed to be 0. Remember to account for any control signals that you added or modified in the previous part of the question!

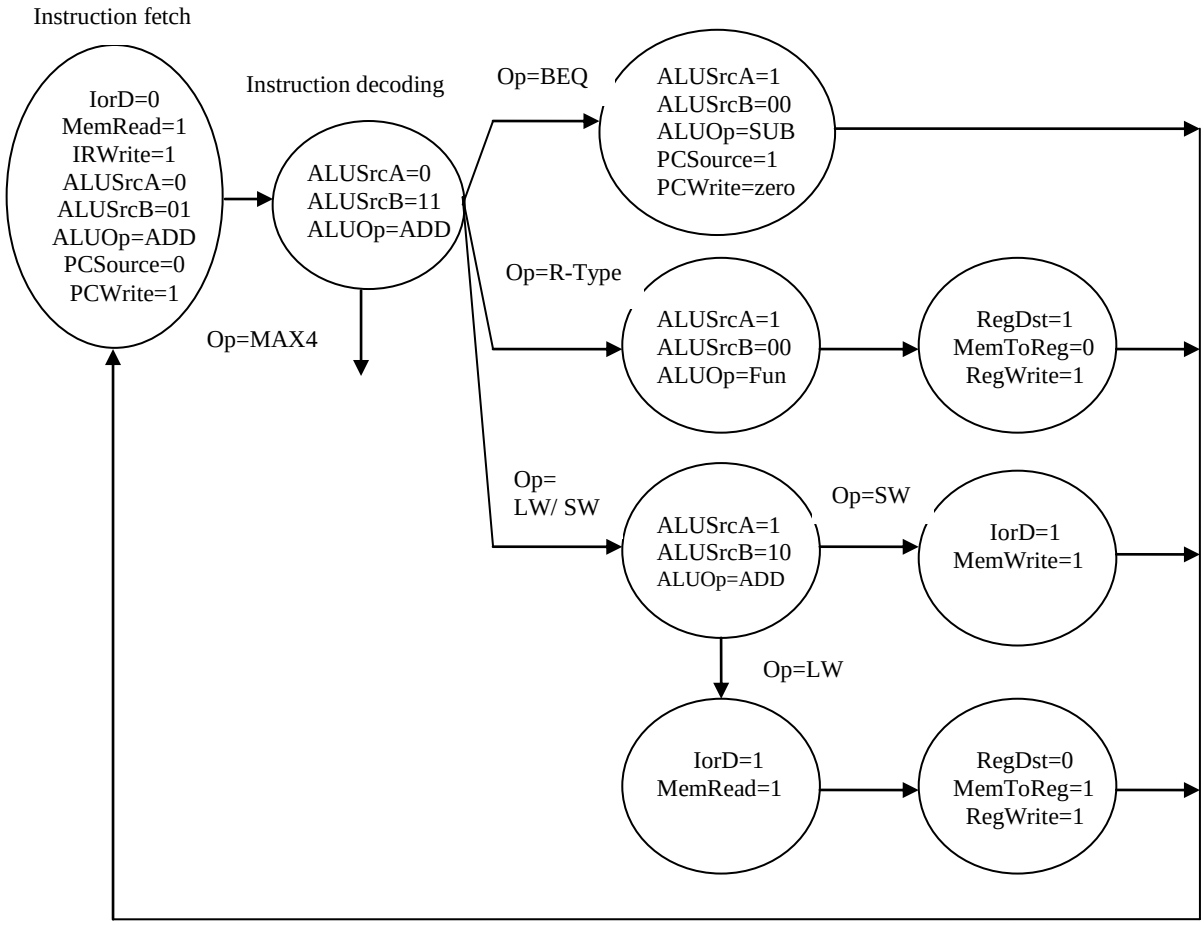


Figure 5.2 The finite state machine of improved MIPS

Question 6 Pipeline Datapath[20 marks]

This question is based on the pipelined datapath shown in Figure 6.1. Consider the following piece of code.

Add $7, $8, $9
Add $6, $7, $5
Add $5, $7, $6
Lw $3, 4($5)

If Add $7, $8, $9 is at WB stage, fill in the correct datapath values for twenty-three question marks ? in the datapath. There is two ? in the IF stage, seven ? in the ID stage, ten ? in EX, two ? in MEM, and two ? in WB.

Again, please :

- Write your answers directly on the diagram on the attached answer sheet.
- Show decimal values.
- Assume that registers initially contain their number plus 100: $5 contains 105, $8 contains 108, etc., and Memory[xx] = xx. All values are decimal.
- Add format: add $Rd, $Rs, $Rt; Lw format: Lw $Rt, offset($Rs).
- Write 'X' for any numbers that can not be determined.

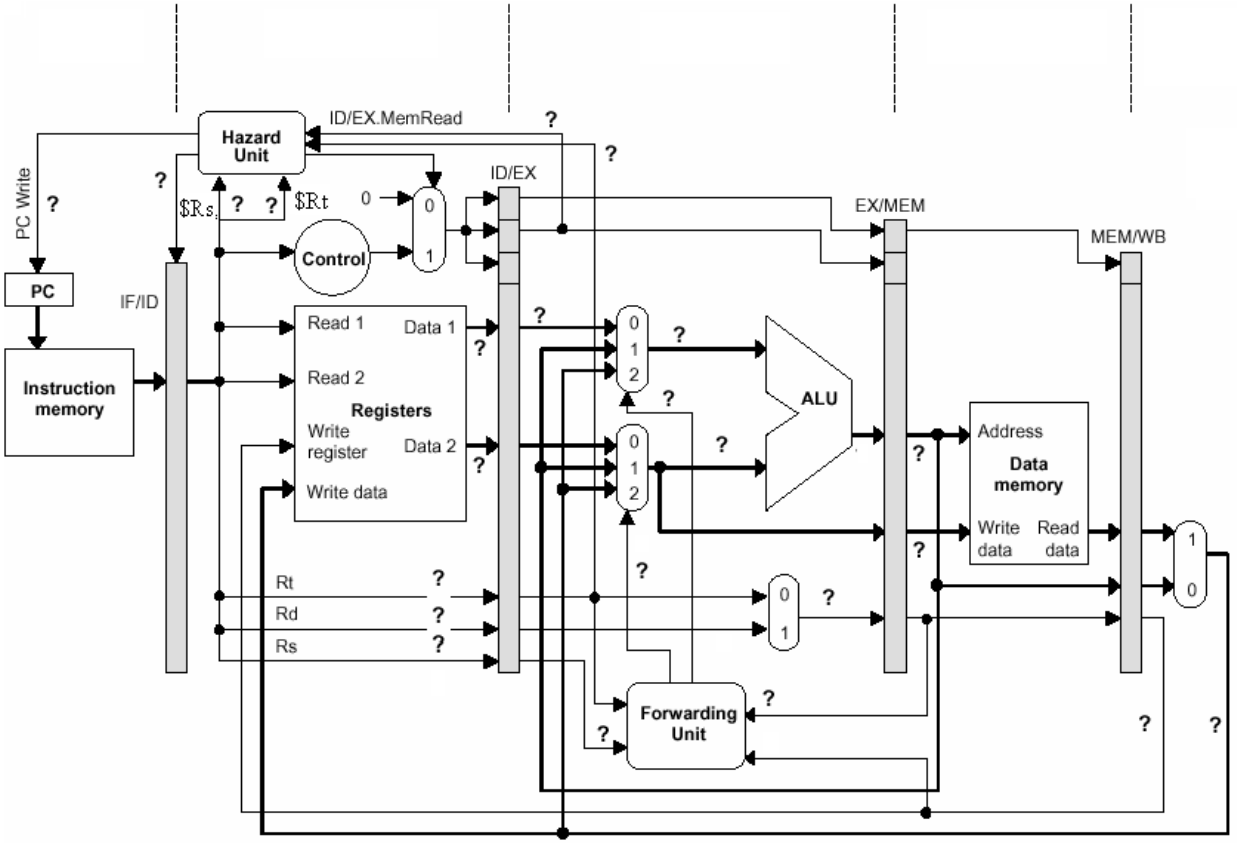


Figure 6.1 The pipelined datapath