Computer Organization and Structure

1. We wish to compare the performance of two different computers: M1 and M2. The following measurements have been made on these computers:

Program	Time on M1	Time on M2
1	2.0 sec.	1.5 sec.
2	5.0 sec.	10.0 sec.

Which computer is faster for each program, and how many times as fast is it?

Consider the two computers and programs as the above. The following additional measurements were made:

Program	Instructions executed on M1	Instructions executed on M2
1	5×10^9	6×10^9

Find the instruction execution rate (instructions per second) for each computer when running program 1. Suppose that M1 costs $500 and M2 costs $800. If you needed to run program 1 a large number of times, which computer would you buy in large quantities? Why? If the clock rates of computers M1 and M2 are 4GHz and 6GHz, respectively, find the clock cycles per instruction (CPI) for program 1 on both computers.

Assuming the CPI for program 2 on each computer as the above is the same as the CPI for program 1 found in the above sub-question, find the instruction count for program 2 running on each computer using the execution times from the first sub-question.

2. Consider two different implementations, P1 and P2, of the same instruction set. There are five classes of instructions (A, B, C, D, and E) in the instruction set. P1 has a clock rate of 4GHz. P2 has a clock rate of 6GHz. The average number of cycles for each instruction class for P1 and P2 is as follows:

Class	CPI on P1	CPI on P2
A	1	2
B	2	2
C	3	2
D	4	4
E	3	4

Assume that peak performance is defined as the fastest rate that a computer can execute any instruction sequence. What are the peak performances of P1 and P2 expressed in instructions per second? If the number of instructions executed in a certain program is divided equally among the classes of instructions except for class A, which occurs twice as often as each of the others, how much faster is P2 than P1?

3. We wish to add the instructions `addi` (add immediate), `jr` (jump register), `sll` (shift left logical), and `lui` (load upper immediate) to the single-cycle datapath. Add any necessary datapaths and control signals to Figure 1 and show the necessary additions to Table 1. You can photocopy Figure 1 and Table 1 to make it faster to show the additions.
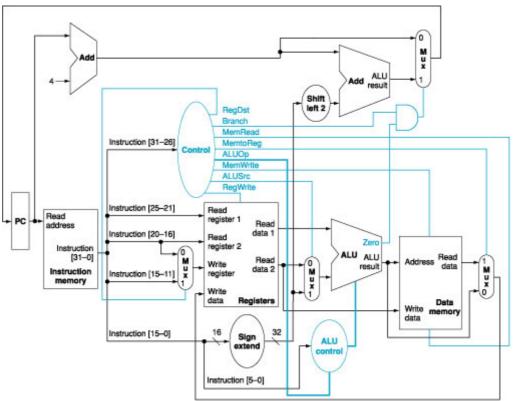

Figure 1: The simple datapath with the control unit.

Instruction	RegDst	ALUSrc	Memto Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOp1	ALUOp0
R-format	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw	X	1	X	0	0	1	0	0	0
beq	X	0	X	0	0	0	1	0	1

Table 1: The setting of the control lines is completely determined by the opcode fields of the instruction.

4. We wish to add the instructions `lui` (load upper immediate) and `ldi` (load immediate) to the multicycle datapath, respectively. The `ldi` instruction loads a 32-bit immediate value from the memory location following the instruction address. Use the same structure of the multicycle datapath of Figure 2 and show the necessary modifications to the finite state machine of Figure 3. How many cycles are required to implement this instruction?
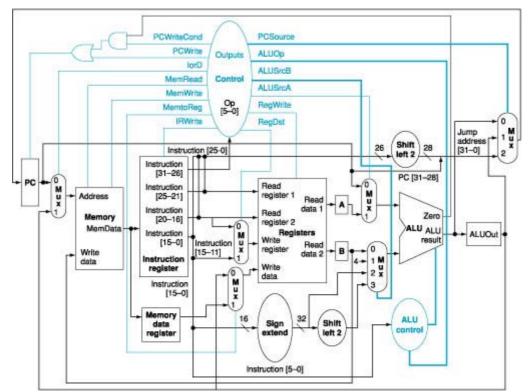
Outputs

PCWriteCond
PCWrite
IorD
MemRead
MemWrite
MemtoReg
IRWrite

Control

Op
[5–0]

PCSource
ALUOp
ALUSrcB
ALUSrcA
RegWrite
RegDst

Instruction [25-0]

Instruction [31–26]
Instruction [25–21]
Instruction [20–16]
Instruction [15–0]
Instruction [15–11]
Instruction [15–0]

PC

Address
Memory
MemData
Write data

Instruction register

Memory data register

Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2

A
B

Sign extend

Shift left 2

ALU control

26
Shift left 2
28

Jump address [31–0]

PC [31–28]

Zero
ALU ALU result

ALUOut

16   32

Instruction [5–0]

Figure 2: The complete datapath for the multicycle implementation together with the necessary control lines.

Instruction fetch

0
MemRead
ALUSrcA = 0
IorD = 0
IRWrite
ALUSrcB = 01
ALUOp = 00
PCWrite
PCSource = 00

Start

Instruction decode/register fetch

1
ALUSrcA = 0
ALUSrcB = 11
ALUOp = 00

(Op = 'LW') or (Op = 'SW')
(Op = R-type)
(Op = BEQ)
(Op = 'J')

Memory address computation

2
ALUSrcA = 1
ALUSrcB = 10
ALUOp = 00

Execution

6
ALUSrcA = 1
ALUSrcB = 00
ALUOp = 10

Branch completion

8
ALUSrcA = 1
ALUSrcB = 00
ALUOp = 01
PCWriteCond
PCSource = 01

Jump completion

9
PCWrite
PCSource = 10

(Op = 'LW')
(Op = 'SW')

Memory access

3
MemRead
IorD = 1

Memory access

5
MemWrite
IorD = 1

R-type completion

7
RegDst = 1
RegWrite
MemtoReg = 0

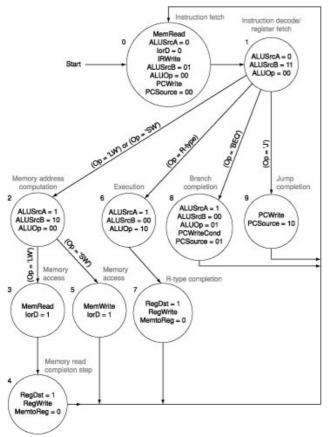Memory read completion step

4
RegDst = 1
RegWrite
MemtoReg = 0

Figure 3: The complete finite state machine control for the datapath shown in Figure 2.

5. In estimating the performance of the single-cycle implementation, we assumed that only the major functional units had any delay (i.e., the delay of the multiplexors, control unit, PC access, sign extension unit, and wires was considered to be negligible). Assume that we use a different type of adder for simple addition:

Memory units: 200ps
ALU: 200ps
adder for PC + 4: Xps
adder for branch address computation: Yps
Register file (read or write): 100ps

   a. What would the cycle time be if X=300 and Y=300?
   b. What would the cycle time be if X=500 and Y=500?
   c. What would the cycle time be if X=100 and Y=800?