Problem 2. Machine M1 has a **single-cycle** implementation. The time required for the execution of each type of instruction is given below, as also the frequency distribution of the instructions for a program P1.

Instruction type	Time to execute an instruction	Frequency distribution for program P1
R-type	6ns	44%
LW	8ns	24%
SW	7ns	12%
Branch	5ns	18%
J	2ns	2%

a. [5 points] What should be the minimum clock cycle time for this machine? Explain.

b. [7 points] If you run a program P1, that has 4539 instructions on M1, what is the CPU execution time for P1 on M1?

c. Extra credit [5 points]
Given the above frequency distribution for the instructions for P1, determine what percent of the CPU execution time, the CPU is idle.

Problem 3. The RTL descriptions for the R-type instructions, memory-reference instructions and flow-control instructions for a MIPS architecture using **multi-cycle** implementation is given below in Table 3.1. Table 3.2 lists the time taken to execute each type of operation (not an entire instruction, just the operation). Table 3.3 lists the frequency distribution for the instructions of program P1.

Table 3.1 RTL descriptions for MIPS instructions

R-Type	sw	lw	beq	j
IR = Mem[PC] PC = PC + 4				
A = Reg[IR[25:21]] B = Reg[IR[20:16]] ALUOut = PC + (SE(IR[15-0]) << 2)				
ALUOut = A op B	ALUOut = A + SE(IR[15-0])		If(A-B==0) then PC = ALUOut	PC = PC[31:28]||IR[25:0]||00
Reg[IR[15:11]] = ALUOut	Mem[ALUOut] = B	MDR = Memory[ALUOut]		
		Reg[IR[20-16]] = MDR		

Operation type	Time to execute the operation
ALU	2 ns
Memory reference	2 ns
Register file access	1 ns

Table 3.2 Time to execute different types of operations

Instruction type	Frequency distribution for instructions of program P1
R-type	44%
LW	24%
SW	12%
Branch	18%
J	2%

Table 3.3 Frequency distribution for the instructions in program P1

a. [5 points] What should be the minimum clock cycle time for this processor? Explain.

b. [10 points] If you were to run P1, which has 4539 instructions and the instruction distribution as shown in Table 3.3, on this processor, what will the CPU execution time be? If you were unable to determine the cycle time in the previous question, assume that the clock rate is 100MHz.

c. Let's relax one of the rules for the RTL. Let us allow a register access operation and a memory access operation to occur sequentially in one clock cycle. This would result in a reduction in the number of clock cycles for one or more MIPS instruction types listed in Table 3.1.
   i. [5 points] Identify which of the instruction types (R-type, LW, SW, BEQ, J) will have a reduction in the number of clock cycles. Do not try to modify clock cycles 1 and 2 for any of the instruction types, even if they can be modified.

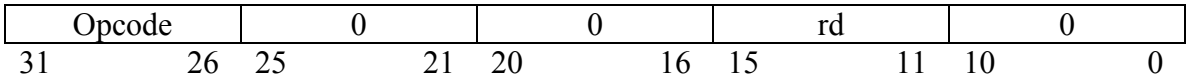
   ii. [5 points] A reduction in the number of clock cycles will result in a corresponding **increase** in the clock cycle time. Determine the new clock cycle time.

Problem 4 Let's introduce two new instructions `pop` and `push` to the MIPS architecture. The RTL description and datapath modifications for the `pop` instruction are provided in the following pages.

The syntax of the `pop` instruction is as follows:

```
pop  rd
```

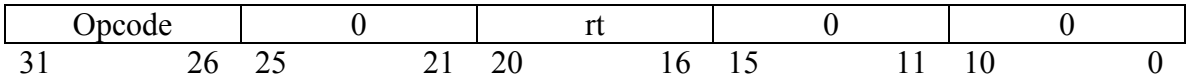
The format is as follows:



The syntax of the `push` instruction is as follows:

```
push rt
```

The format is as follows:



- The stack pointer is the general purpose register $sp ($29) in the MIPS architecture. It is part of the register file.
- The stack pointer points to the top of the stack(TOS).
- The stack is a part of memory. When you access the stack, you are, in fact, accessing the portion of memory reserved for the stack.
- The `pop` instruction copies the value on the top of the stack into register rd. It then moves the stack pointer to point to the new TOS by adding 4 to it(not subtracting).
- The `push` instruction copies the value in register rt onto the stack (on the top). Before it does that, it subtracts 4 from the stack pointer, so that it points to the new TOS.

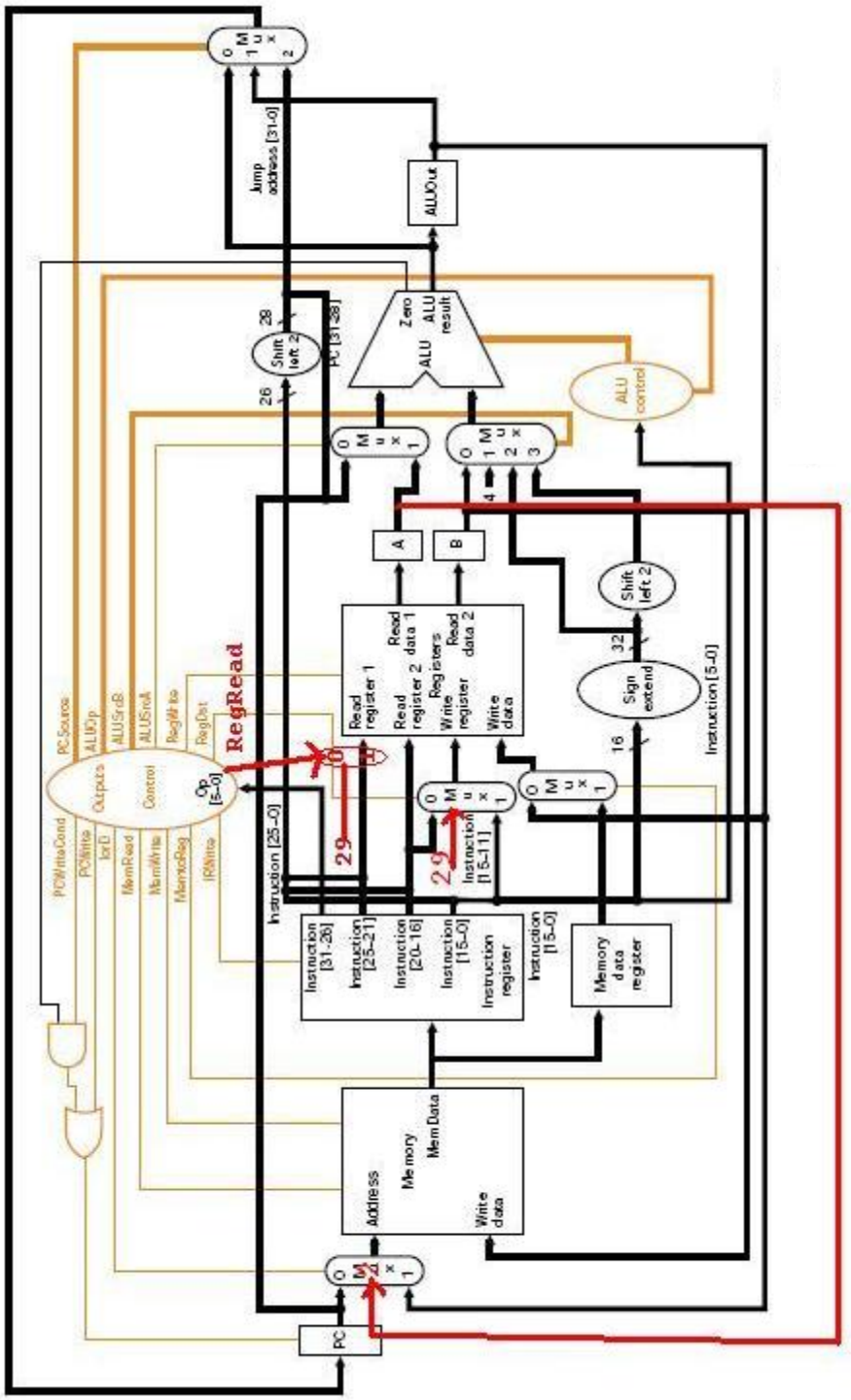
a. [8 points]In the space provided, write the RTL for the `push` instruction. You are provided with the RTL description for the `pop` instruction.

Action for `pop`	Action for `push`
IR = Memory[PC] PC = PC + 4	
A = Reg [IR[25-21]] B = Reg [IR[20-16]] ALUOut = PC + (sign-extend (IR[15-0]) << 2) If (IR[…] = …) then	
A = Reg[29] #Retrieve the value of $sp # i.e. the address of the TOS	
MDR = Mem[A] # Read the value # from the TOS and # place in register MDR.	

Reg[IR[15:11]] = MDR # Write to register
# rd the value from TOS. 	
ALUOut = A + 4 # Ad ubtract 4 from the $sp	

Reg[29] = ALUOut # Update the Register file # with the new value for $sp.	

The datapath modifications for the pop operation are shown below.



b. [8 points] Modify the FSM diagram to implement the pop instruction.