CS152: Section 1

Q1. Architecture vs Microarchitecture

True or **false**: The following is architecturally visible (exposed by the architecture)?

1. Register file entries in a classical RISC pipeline
2. The stack in a stack architecture
3. Pipeline registers
4. Branch-delay / load-delay slots
5. NOPs
6. Pipeline bubbles
7. Condition codes, status flags
8. Memory address width
9. Instruction/data caches

Q2. Microcoded vs Pipelined

1. How does a microcoded machine differ from a classic RISC pipeline?

2. Why is a simpler microarchitecture generally possible with microcoding?

Q3. Microprogramming

Implement a conditional memory-to-memory move instruction in microcode for the single-bus RISC-V machine described in Handout #1. The instruction has the following format:

```
CMOVM (rd), (rs1), rs2
```

CMOVM performs the following operation: If the value in rs2 is true (non-zero), then the memory word loaded from the address in rs1 is stored to the address in rd.

```
if R[rs2] != 0
      M[rd] := M[rs1]
```

Fill in the following table with the microinstructions and control signals. Optimize your microprogram to minimize the number of cycles and to set entries to don't-cares (*) wherever possible.

State	Pseudocode	ldIR	Reg Sel	Reg Wr	en Reg	ldA	ldB	ALUOp	en ALU	ld MA	Mem Wr	en Mem	Imm Sel	en Imm	μBr	Next State
FETCH0	MA := PC; A := PC	*	PC	0	1	1	*	*	0	1	0	0	*	0	N	*
	IR := Mem	1	*	0	0	0	*	*	0	0	0	1	*	0	S	*
	PC := A + 4	0	PC	1	0	0	*	INC_A_4	1	*	0	0	*	0	D	*
...																
NOP0	μBr to FETCH0	*	*	0	0	*	*	*	0	*	0	0	*	0	J	FETCH0
CMOVM0:																