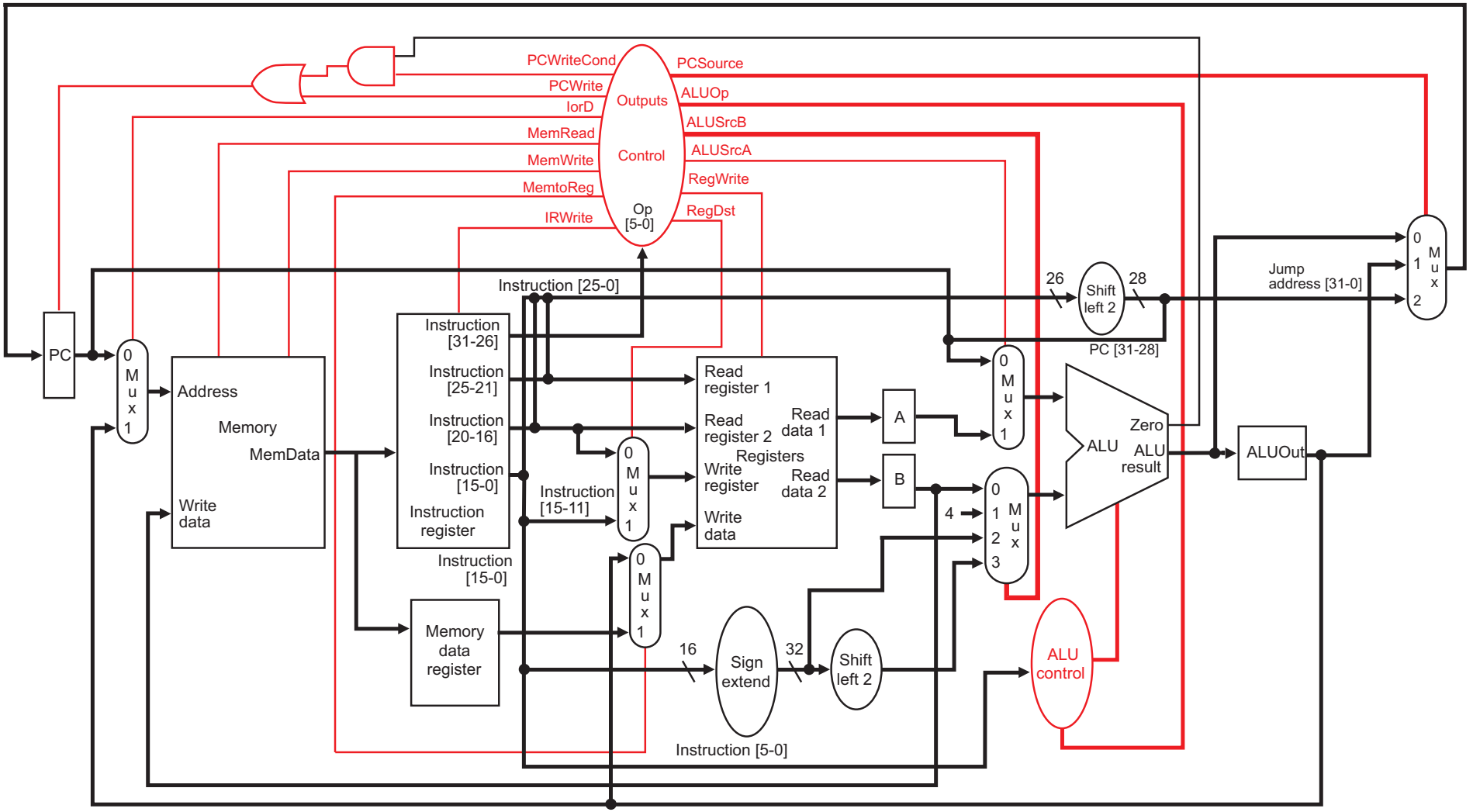


Complete *multi–cycle* architecture

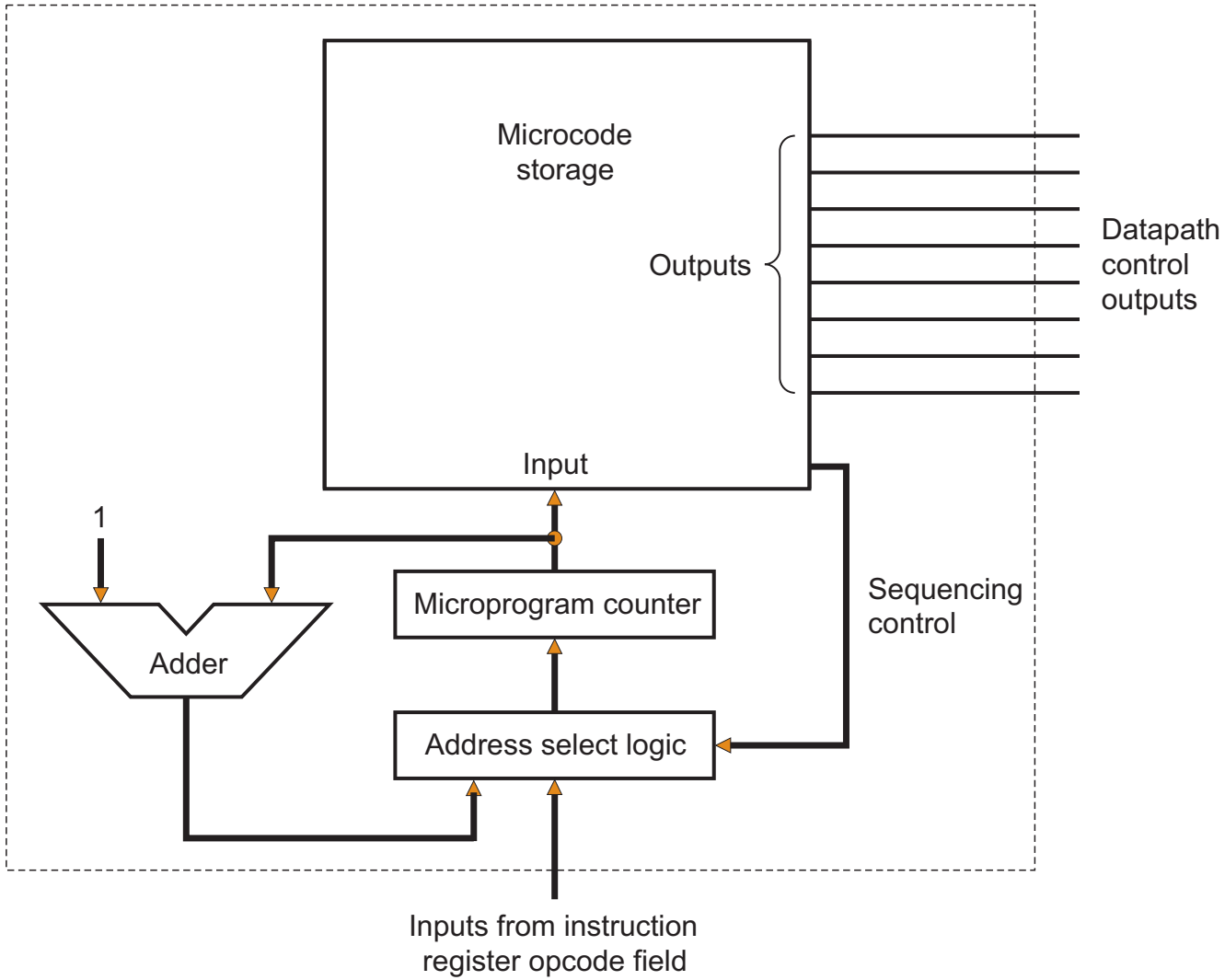
- Microprogram-based control is a ROM organized in n-bit words, called micro-instructions, representing control signals and addressed by microinstruction counter.

- Microinstructions read from the memory in consecutive clock cycles provide the proper execution of each processor instruction.

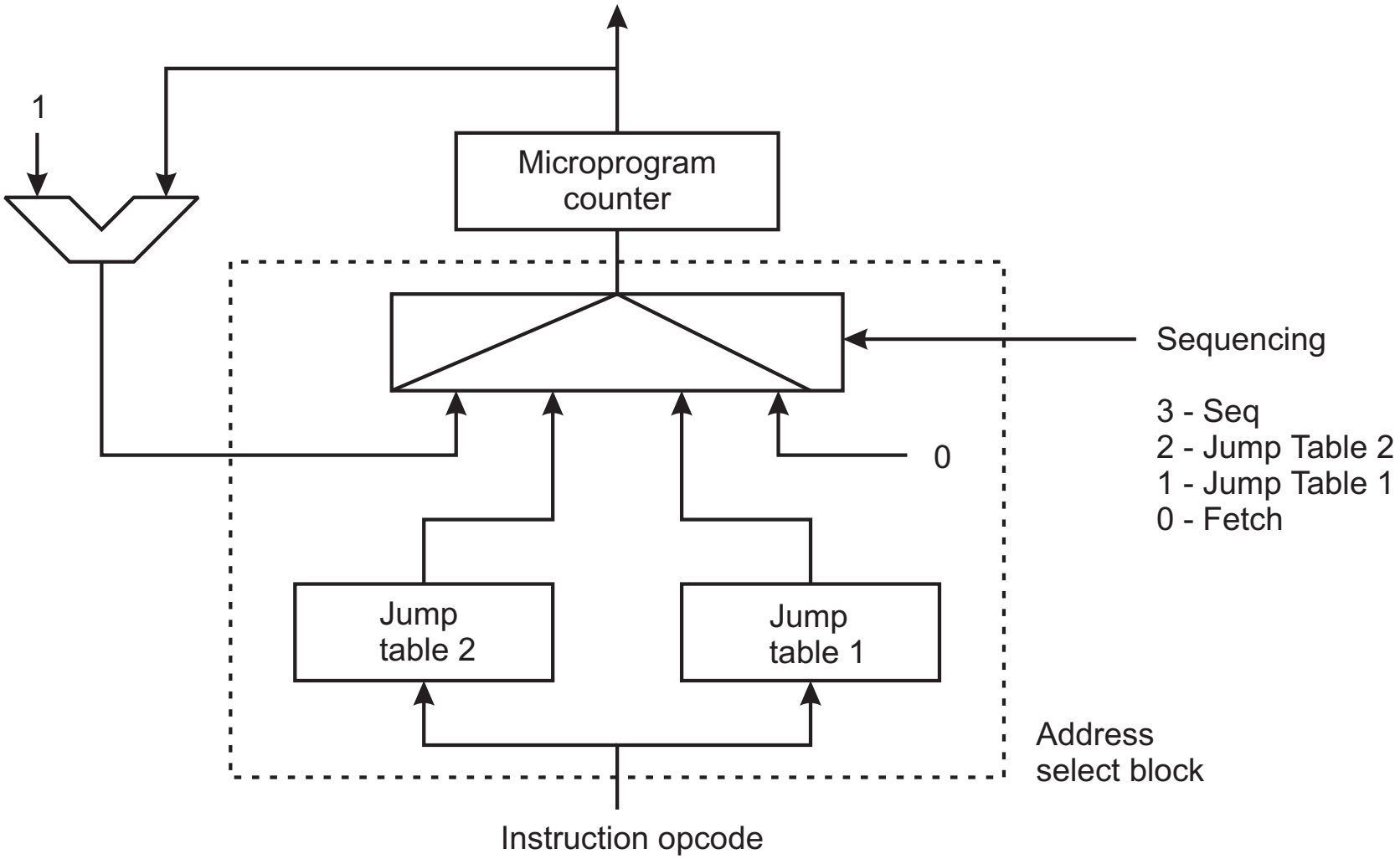
- All the sequences of microinstructions that implement all the processor instructions are called a microprogram.

- Some microinstructions can be shared by various processor instructions, so the size of microprogram memory can be reduced to the number of unique states of the state machine.

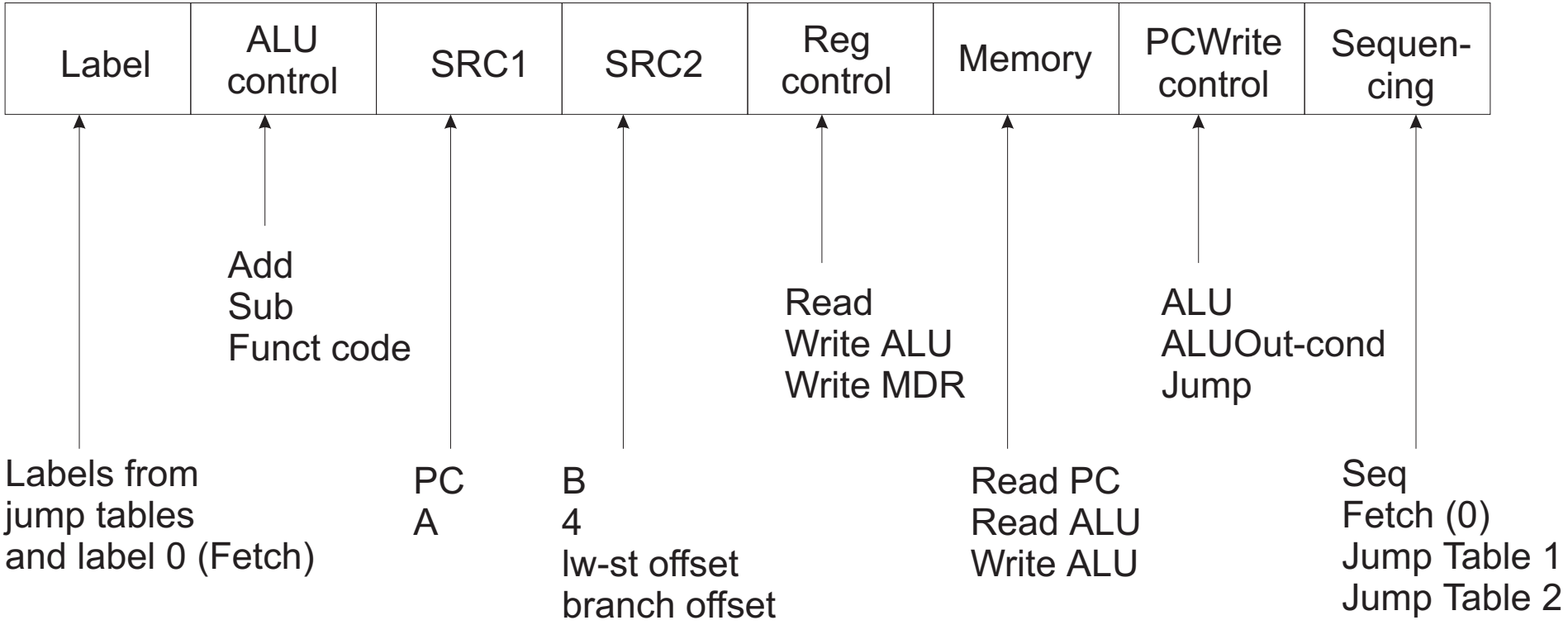
- Microprogram can be executed in sequence or with conditional or unconditional jumps, depending on the instruction opcode.



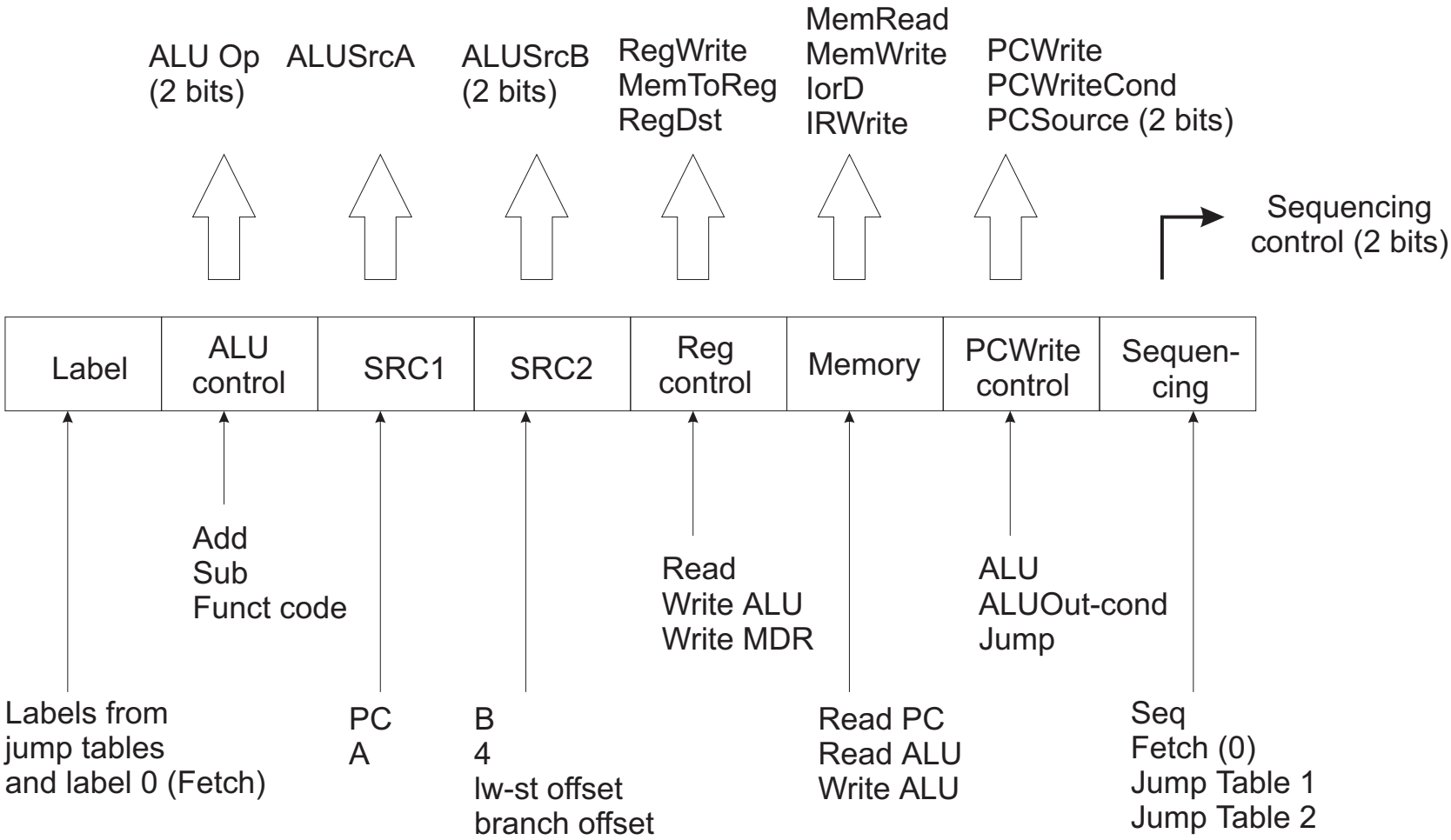
Microprogrammed control unit



Microprogram sequence control — *Sequencing*



Single line of microprogram



Microcode fields as control signals

Label	ALU control	SRC1	SRC2	Reg control	Memory	PCWrite control	Sequen-cing
Fetch	Add	PC	4		Read PC	ALU	Seq
	Add	PC	branch offset	Read			Jump Table1

Fetch and *Decode* microprogram

Label	ALU control	SRC1	SRC2	Reg control	Memory	PCWrite control	Sequen-cing
Fetch	Add	PC	4		Read PC	ALU	Seq
	Add	PC	branch offset	Read			Jump Table1
Mem1	Add	A	ls-sw offset				Jump Table2
LW2					ReadALU		Seq
				WriteMDR			Fetch
SW2					WriteALU		Fetch
Rtype1	Func cod	A	B				Seq
				WriteALU			Fetch
Beq1	Sub	A	B			ALUOut cond.	Fetch
Jump1						Jump	Fetch

Complete microprogram for *multi–cycle* architecture

Jump Table 1

Opcode	Label
Rtype	Rtype1
Jump	Jump1
Beq	Beq1
Load	Mem1
Store	Mem1

Jump Table 2

Opcode	Label
Load	LW2
Store	SW2

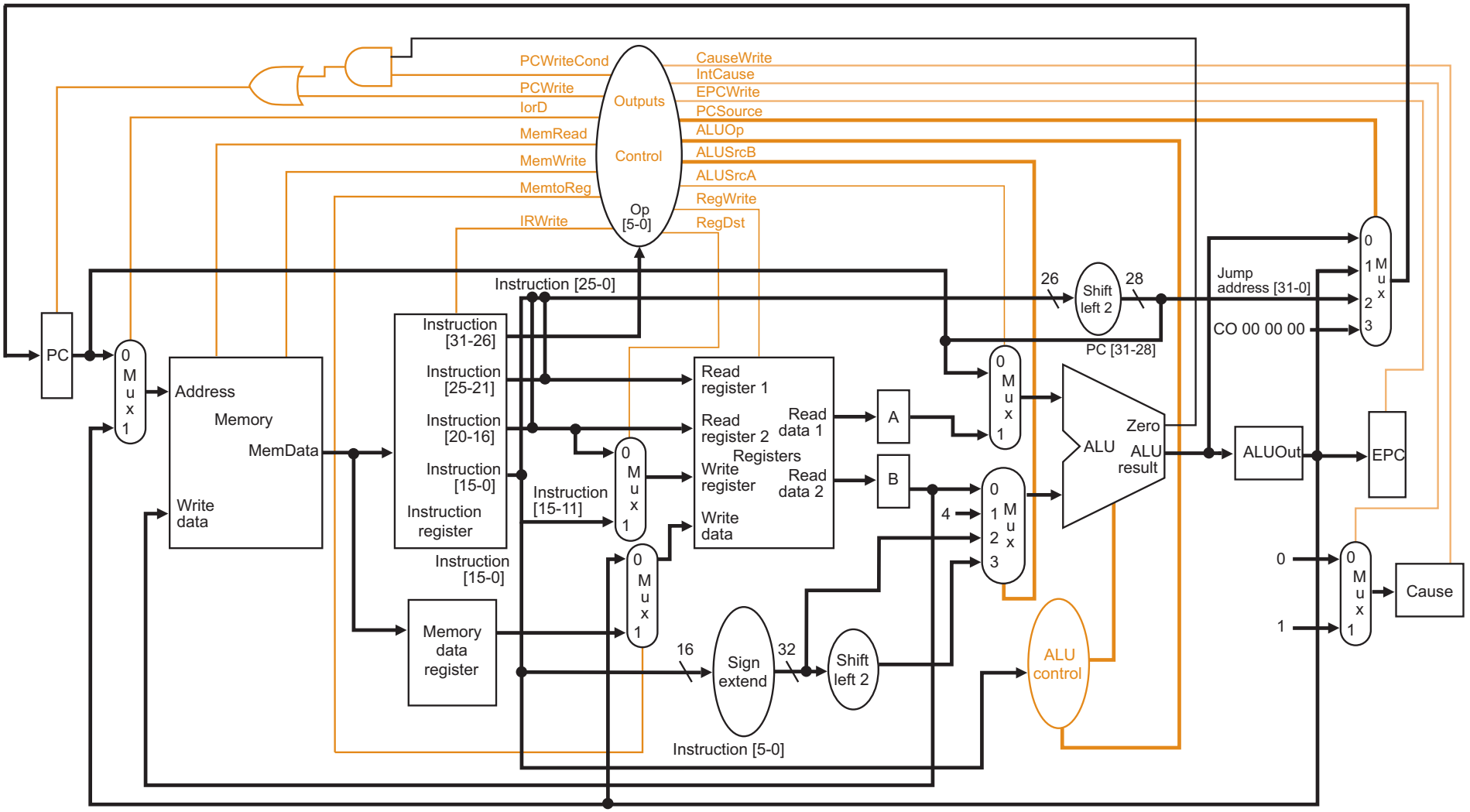
Microprogram jump tables

Exceptions (Interrupts)

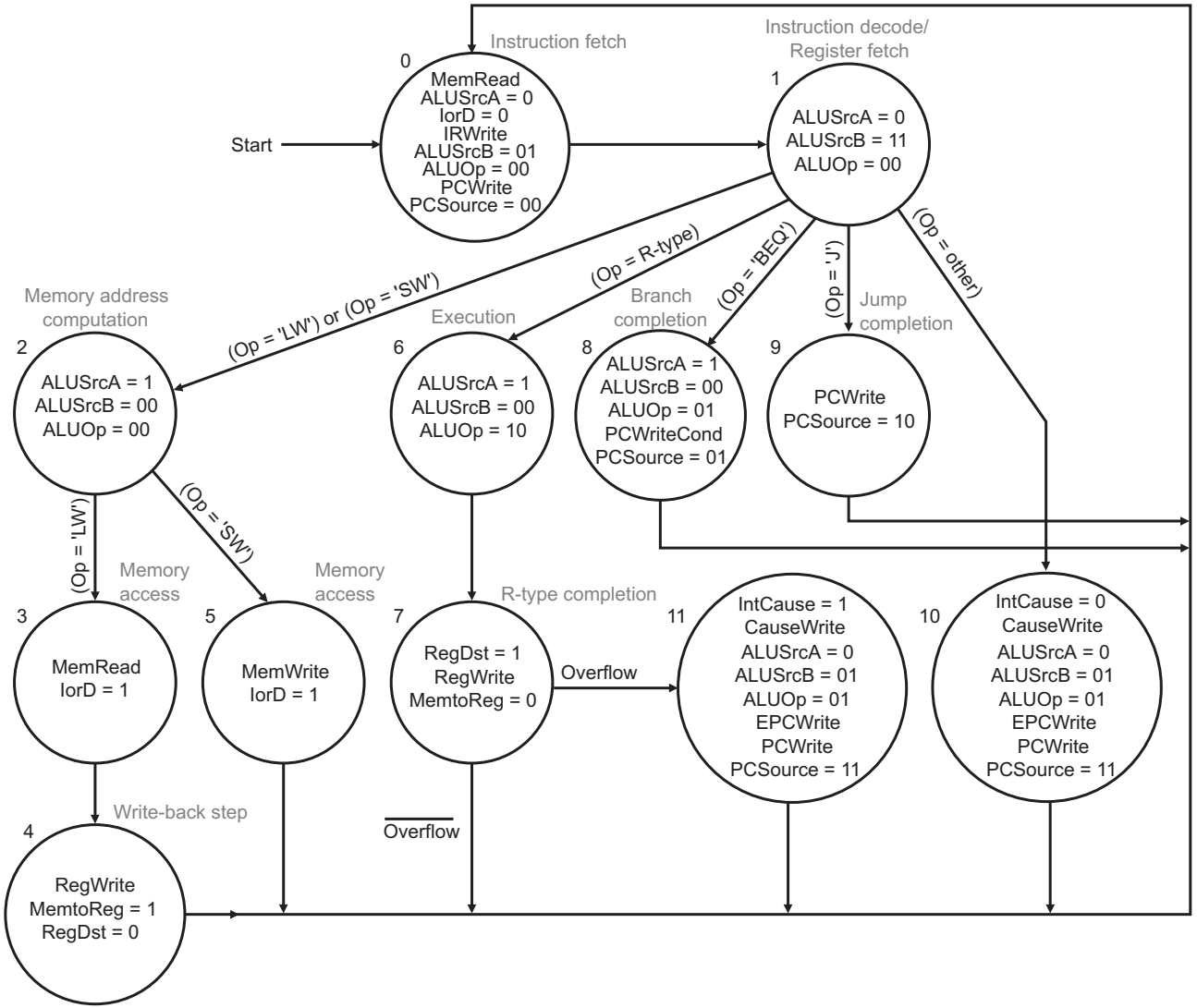
- Exception — an occurrence of an (unpredictable) event at an unpredictable moment requiring an immediate change of normal program sequence (call to an exception handling routine)
- Exceptions can be handled in two ways:
  - jump to an address of individual handling routine (ver. 1)
  - jump to a common address of general handling routine (ver. 2)
- Exception handling mechanism (ver. 2):
  - register EPC (*Exception Program Counter*)
  - register Cause — internal number of exception
  - control lines *CauseWrite*, *IntCause*, *EPCWrite*

Implementation example of exception handling:

- illegal instruction — Cause=0 — an attempt to execute of a bit pattern not corresponding to any defined opcode
- arithmetic overflow — Cause=1 — generation of overflow bit during arithmetic operation in ALU
- address of common exception handling routine `$C0000000` to PC



Implementation of exception handling in *multi-cycle* architecture



Control state-machine with exception handling