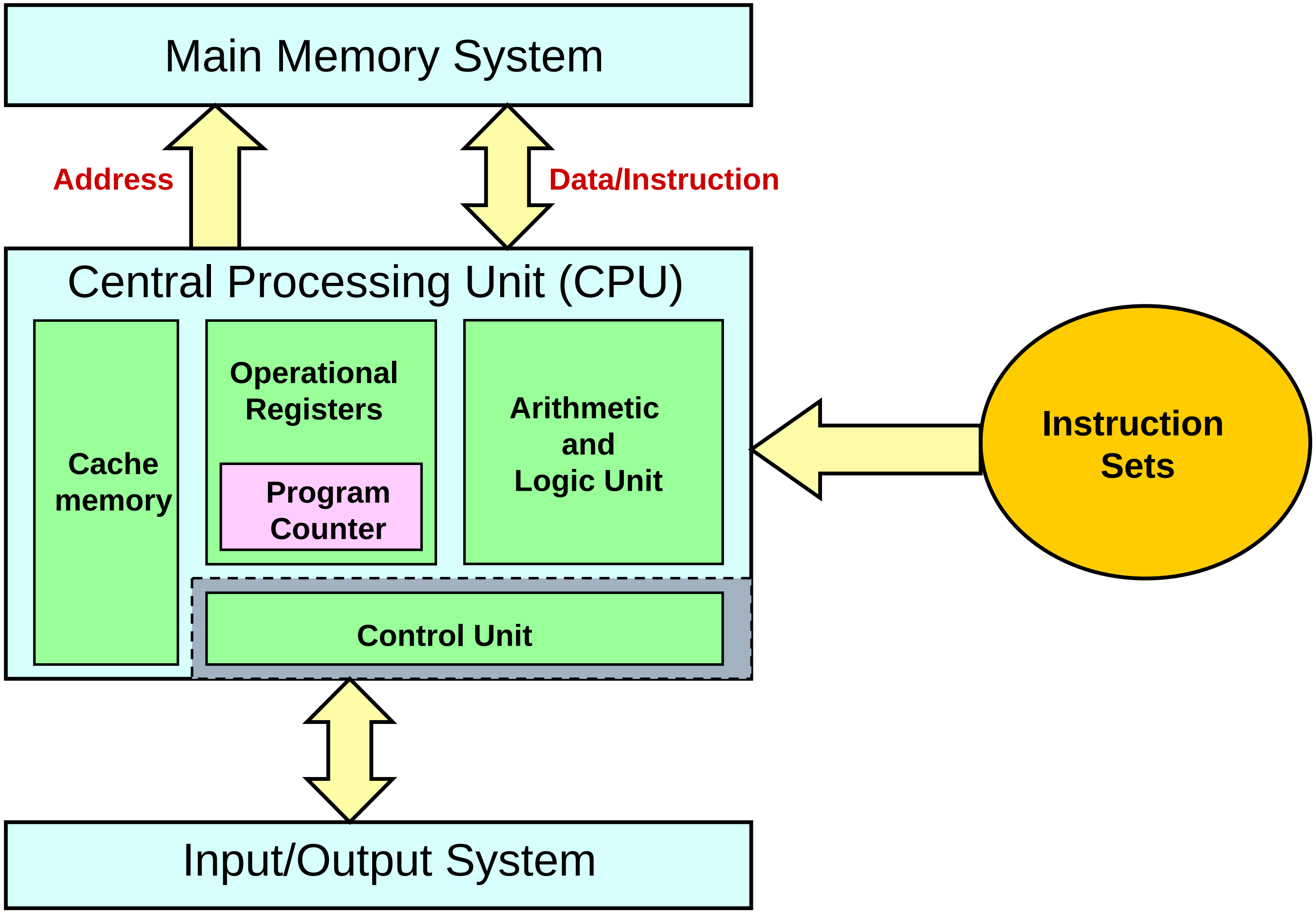
Chapter 8
Basic Processing Unit

Jin-Fu Li
Department of Electrical Engineering
National Central University
Jungli, Taiwan

Outline

- Fundamental Concepts
- Hardwired Control
- Microprogrammed Control

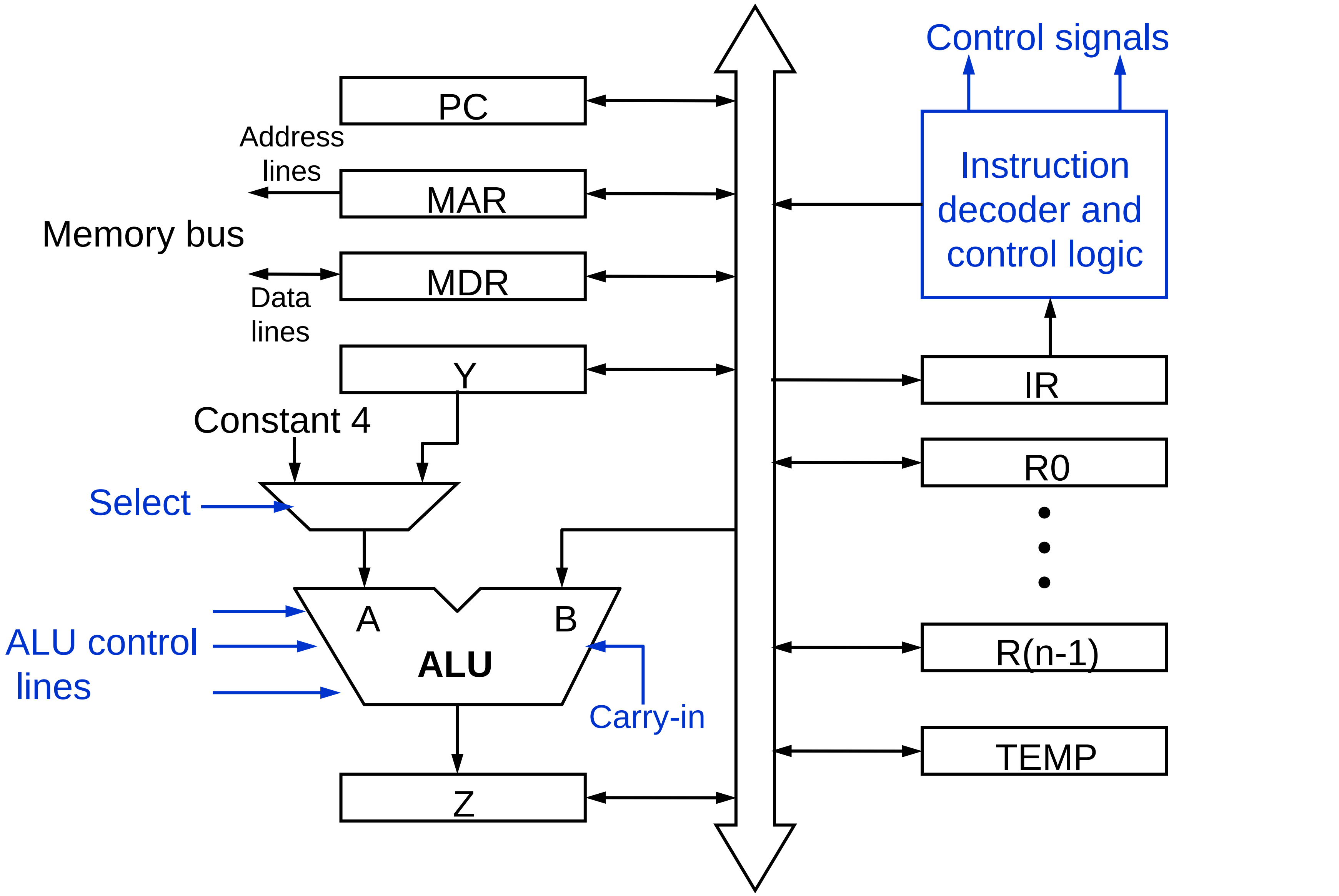
Content Coverage



Instruction Execution

- To execute an instruction, the processor has to perform the following three steps:
  - Step 1: fetch the contents of the memory location pointed by the PC. The contents of this location are interpreted as an instruction to be executed. Hence, they are loaded into the IR. Symbolically, this can be written as IR$\leftarrow$[[PC]]
  - Step 2: assuming that the memory is byte addressable, increment the contents of the PC by 4, that is, PC$\leftarrow$[PC]+4
  - Step 3: carry out the actions specified by the instruction in the IR
- Step 1 and Step 2$\rightarrow$*fetch phase*
- Step 3$\rightarrow$*decode phase, execution phase, and/or write phase*

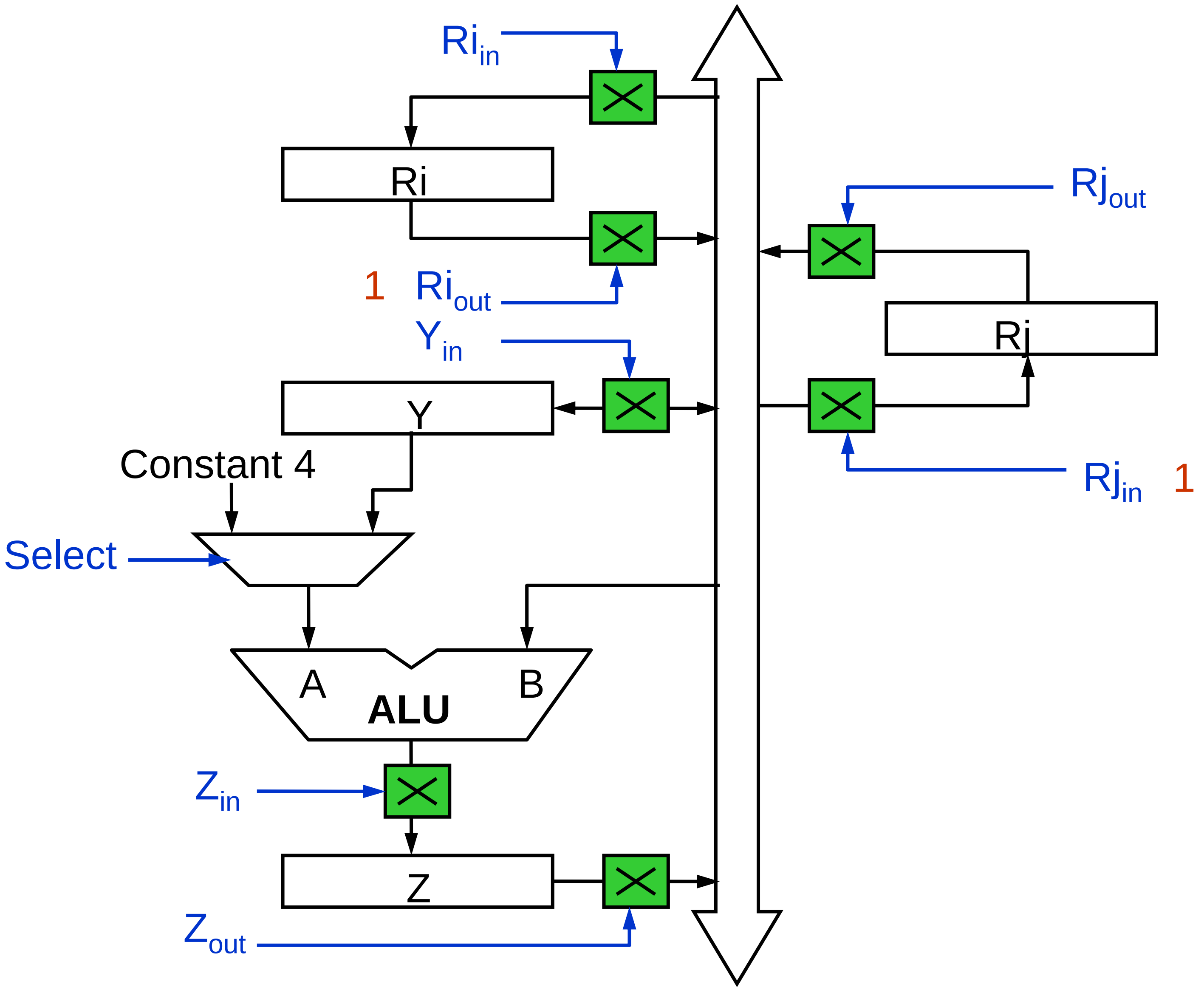
Single-Bus Organization of the Datapath



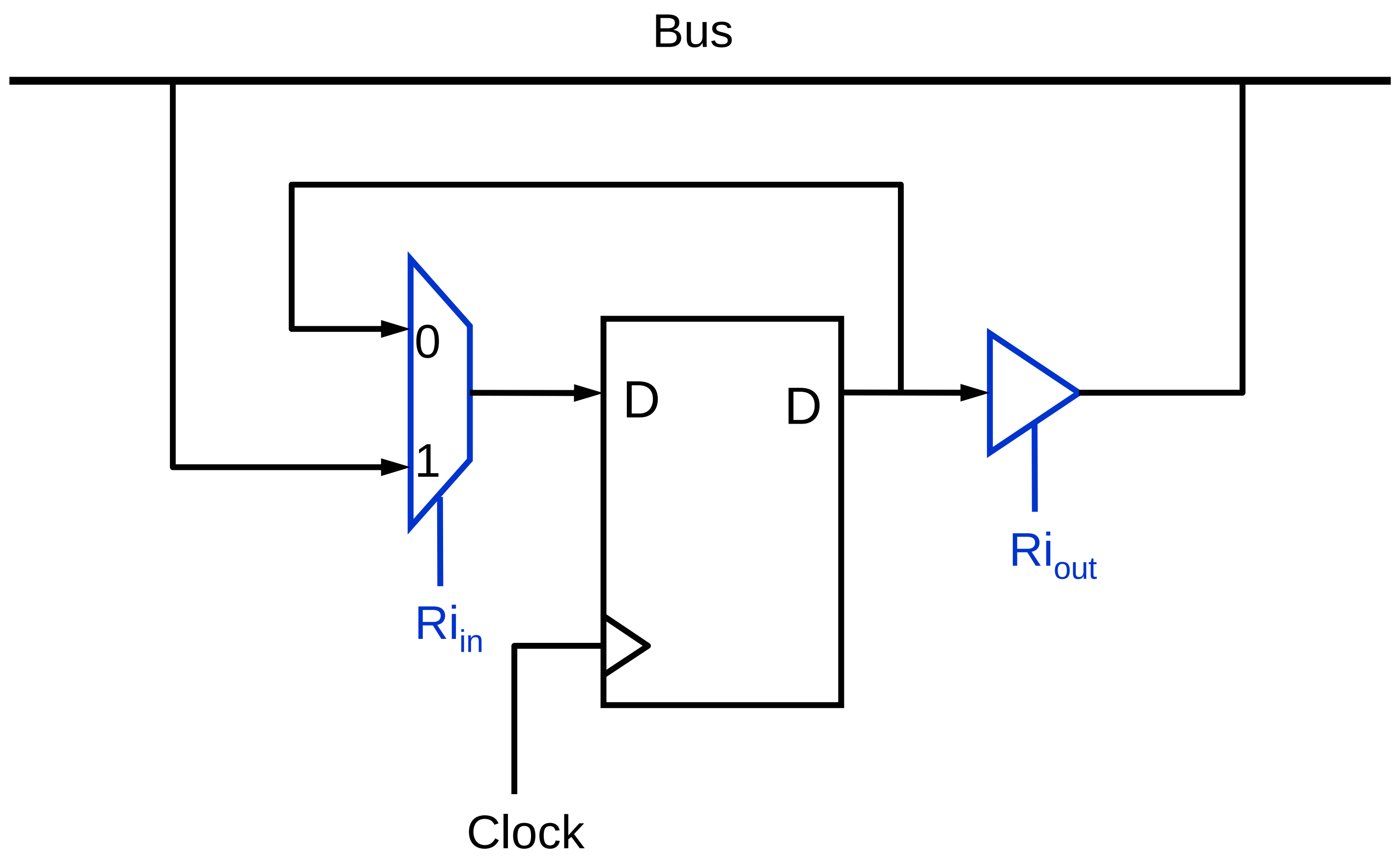
Instruction Execution

- An instruction can be executed by performing one or more of the following operations in some specified sequence
  - Transfer a word of data from one processor register to another or to the ALU
  - Perform an arithmetic or a logic operation and store the result in a processor register
  - Fetch the contents of a given memory location and load them into a processor register
  - Store a word of data from a processor register into a given memory location

Register Transfers

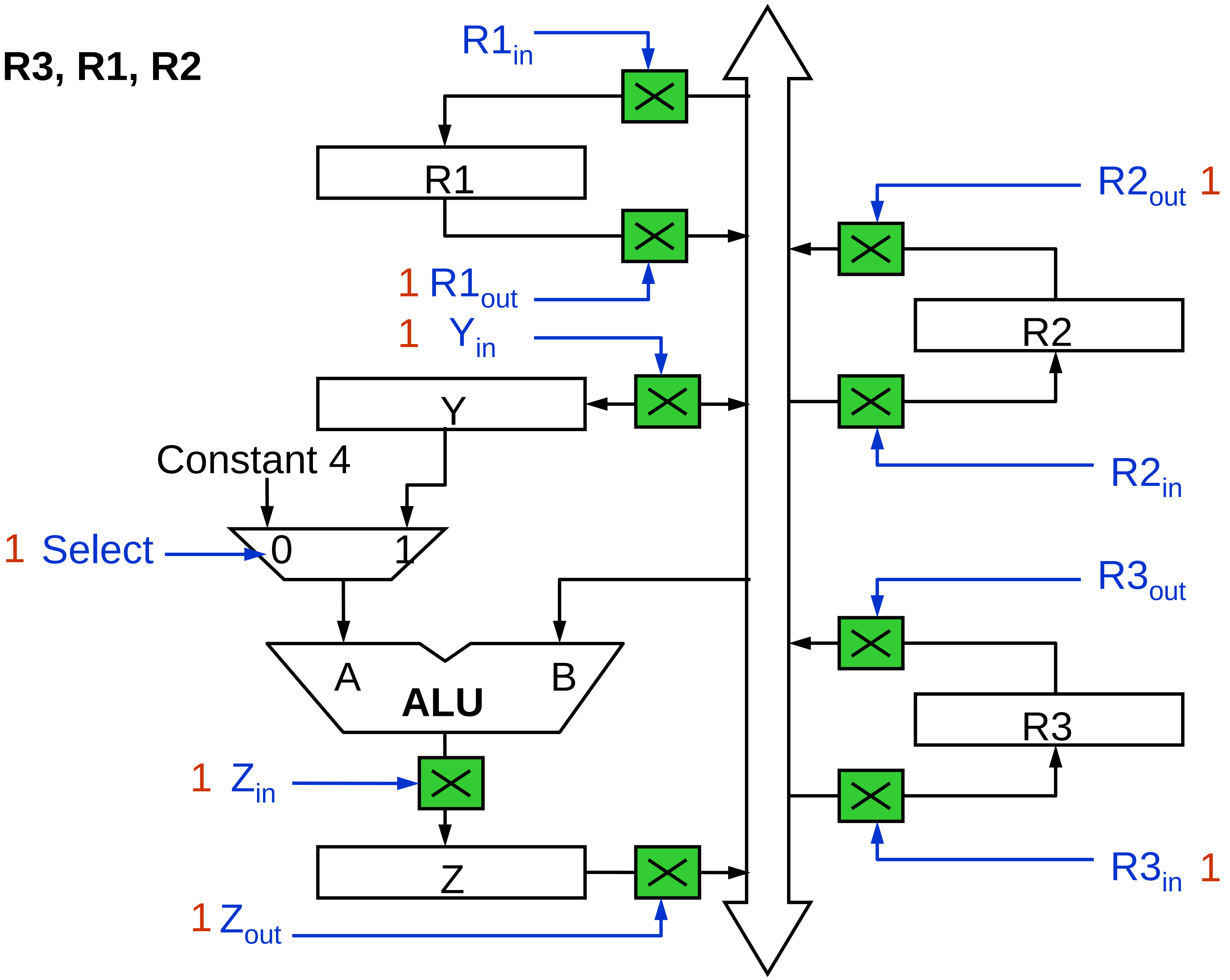


Input and Output Gating for 1-bit Register

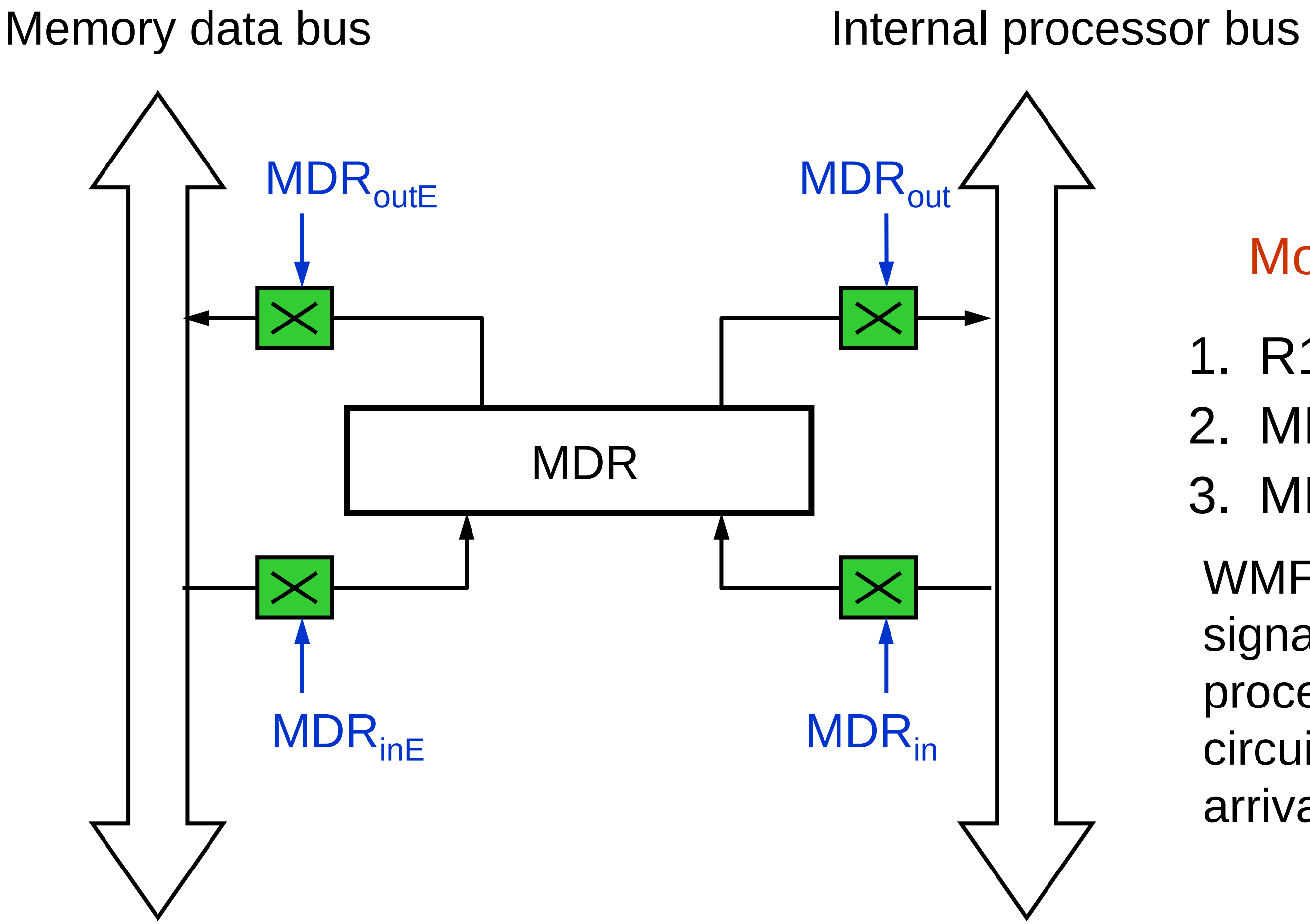


Arithmetic Operation

Add R3, R1, R2



Fetching a Word from Memory



Move (R1), R2

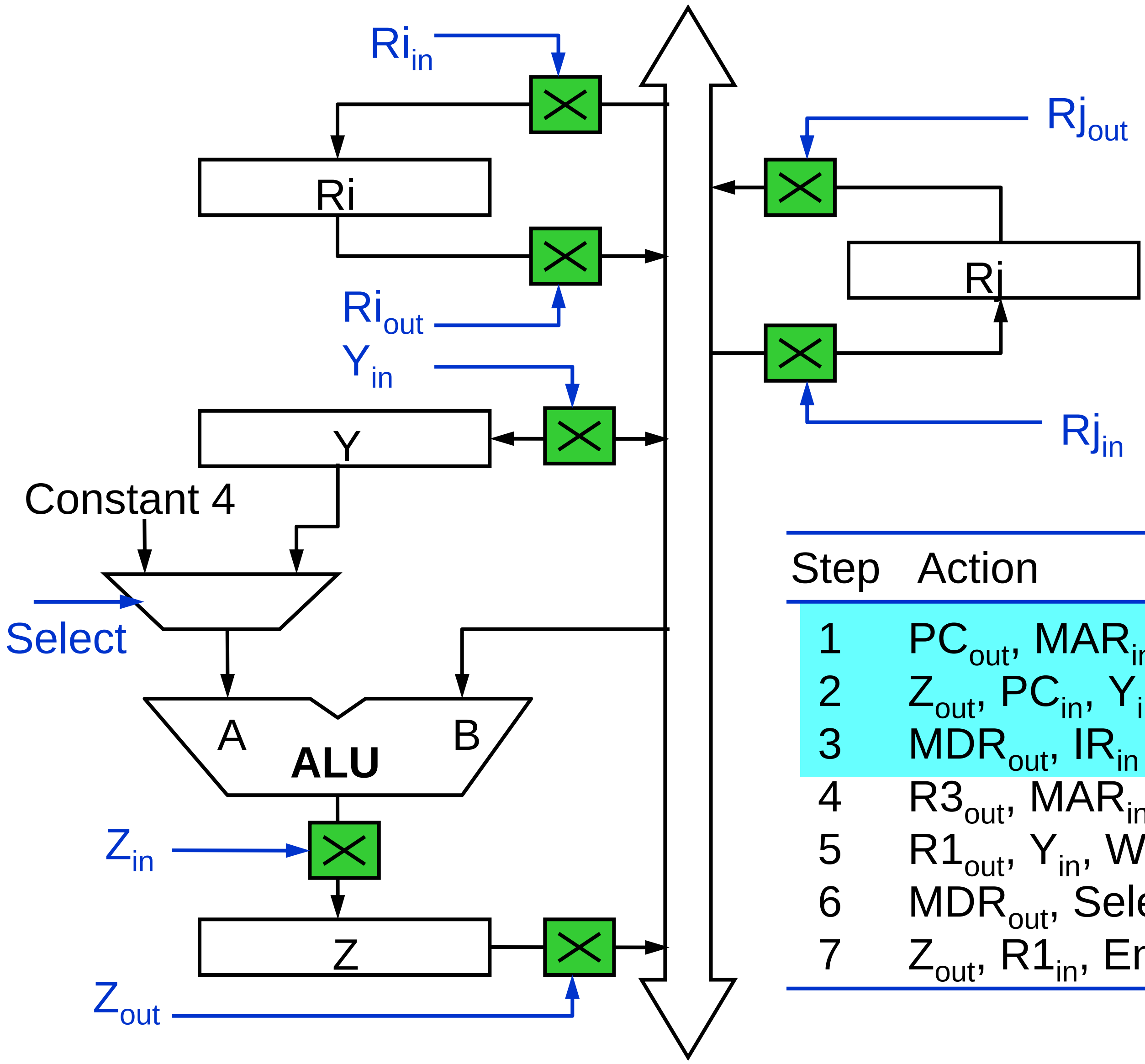
1. $R1_{out}$, MARin, Read
2. MDRinE, WMFC
3. MDRout, R2in

WMFC: is the control signal that causes the processor's control circuitry to wait for the arrival of the MFC signal.

Execution of a Complete Instruction

- Ass (R3), R1: adds the contents of a memory location pointed to by R3 to register R1. Executing this instruction requires the following actions:
  - Fetch the instruction
  - Fetch the first operand (the contents of the memory location pointed to by R3)
  - Perform the addition
  - Load the result into R1

Control Sequence



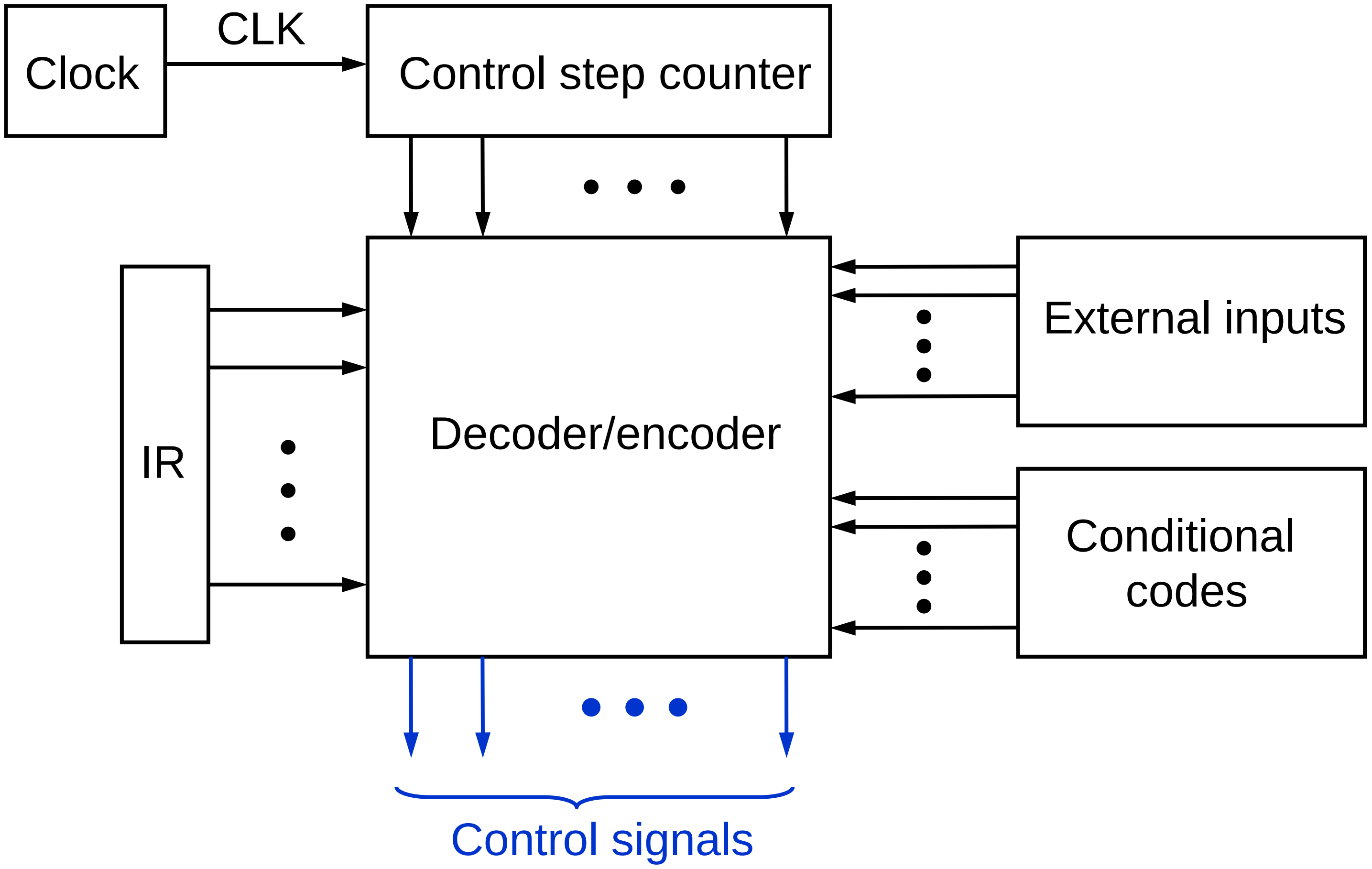
Instruction fetch

Step	Action
1	PC_{out}, MAR_{in}, Read, Select4 Add, Z_{in}
2	Z_{out}, PC_{in}, Y_{in}, WMFC
3	MDR_{out}, IR_{in}
4	$R3_{out}$, MAR_{in}, Read
5	$R1_{out}$, Y_{in}, WMFC
6	MDR_{out}, SelectY, Add, Z_{in}
7	Z_{out}, $R1_{in}$, End

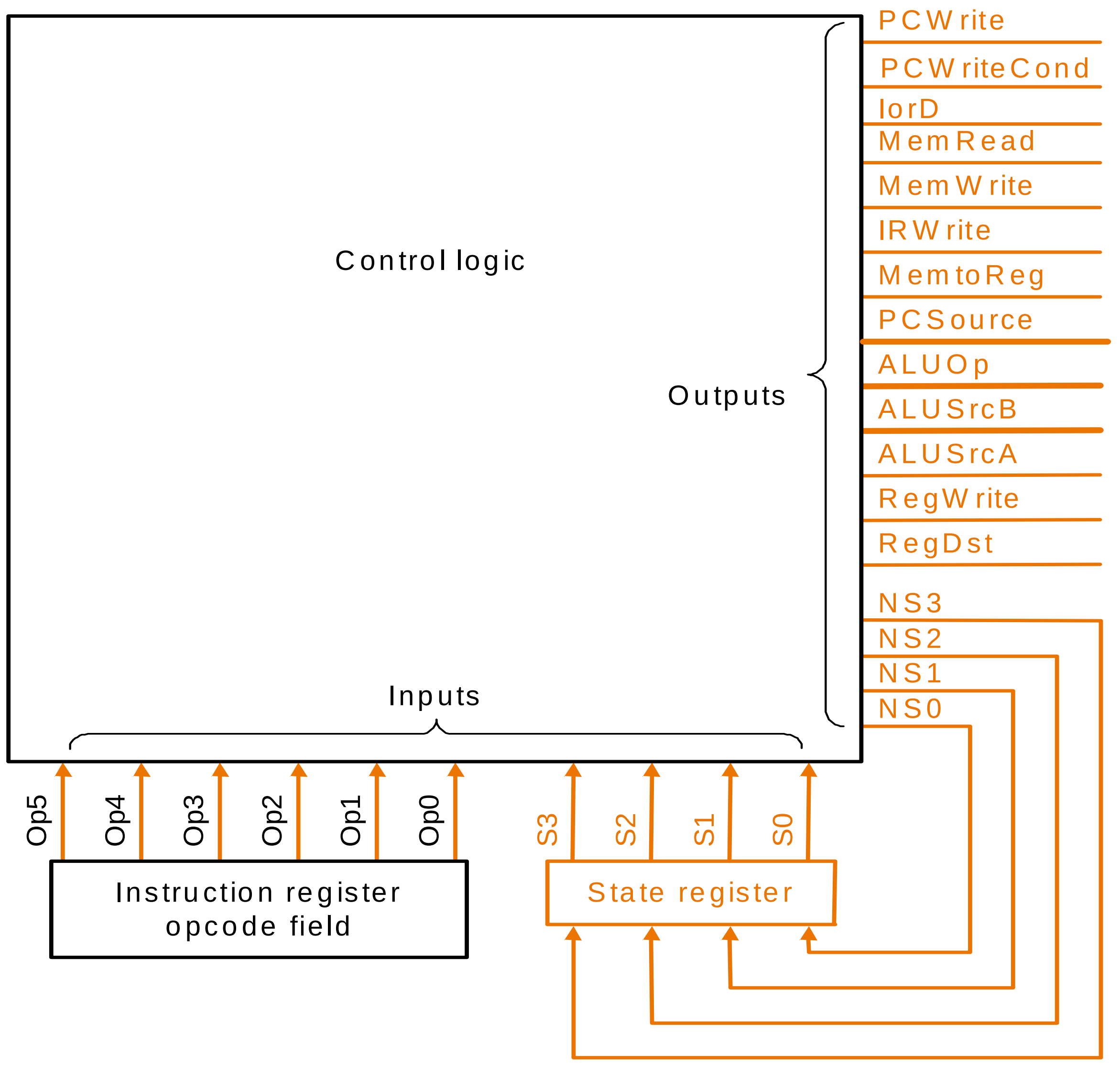
Control

- To execute instructions, the processor must have some means of generating the control signals needed in the proper sequence. Computer designers use a wide variety of techniques to solve this problem.
- The approaches used fall into one of two categories: hardwired control and microprogrammed control

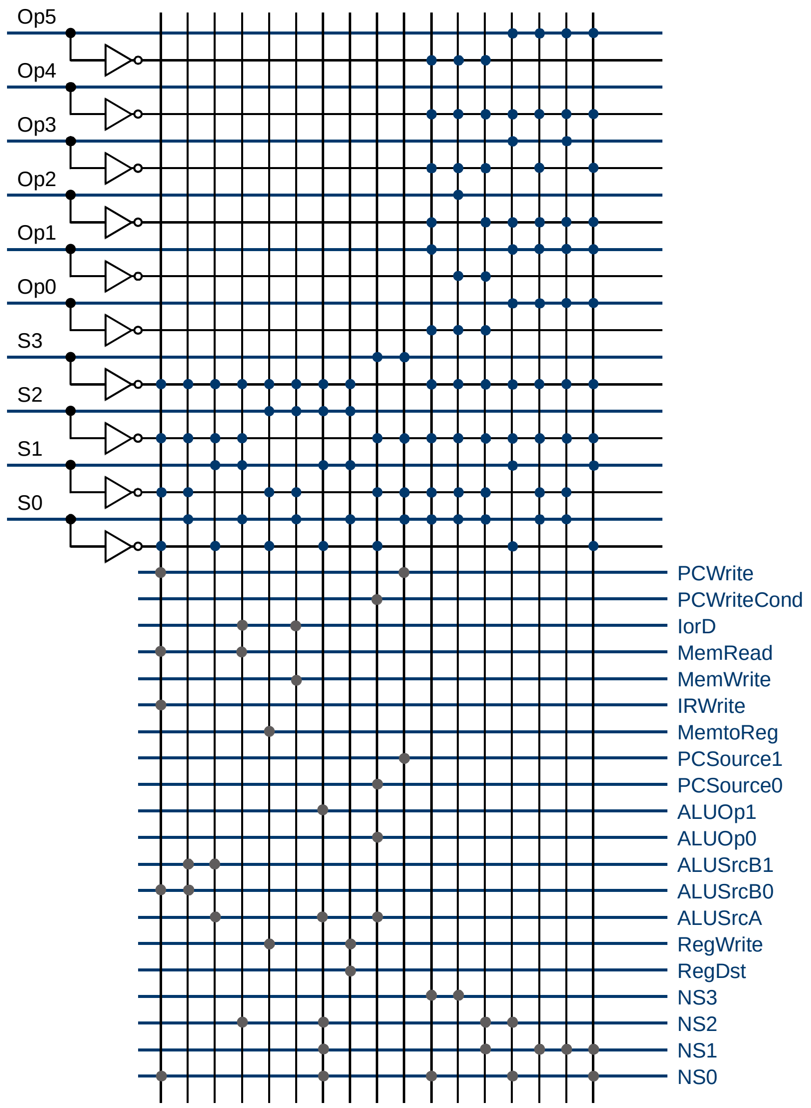
Control Unit Organization



Finite State Machine for Control

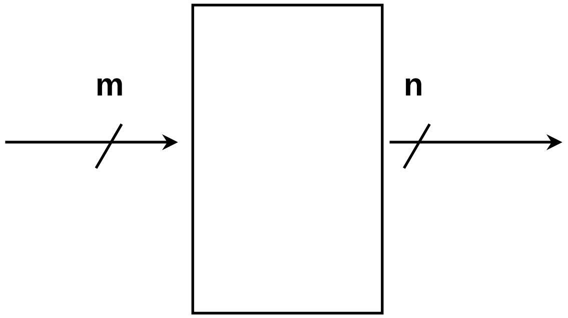


PLA Implementation



ROM Implementation

- ROM = "Read Only Memory"
  - values of memory locations are fixed ahead of time
- A ROM can be used to implement a truth table
  - if the address is m-bits, we can address 2^m entries in the ROM.
  - our outputs are the bits of data that the address points to.

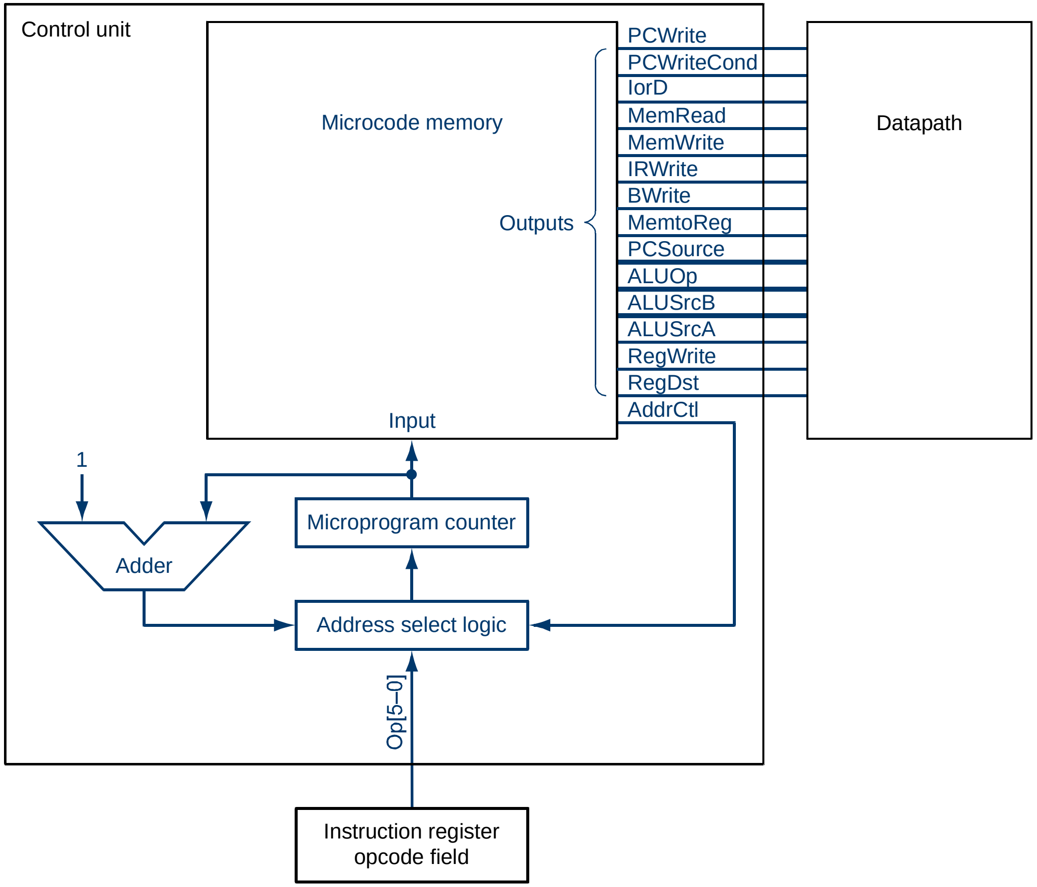


0	0	0	0	0	1	1
0	0	1	1	1	0	0
0	1	0	1	1	0	0
0	1	1	1	0	0	0
1	0	0	0	0	0	0
1	0	1	0	0	0	1
1	1	0	0	1	1	0
1	1	1	0	1	1	1

m is the "height", and n is the "width"

Microprogrammed Control

- Microprogrammed control
  - Control signals are generated by a program similar to machine language programs



Microinstruction format

Field name	Value	Signals active	Comment
	Add	ALUOp = 00	Cause the ALU to add.
ALU control	Subt	ALUOp = 01	Cause the ALU to subtract; this implements the compare for branches.
	Func code	ALUOp = 10	Use the instruction's function code to determine ALU control.
SRC1	PC	ALUSrcA = 0	Use the PC as the first ALU input.
	A	ALUSrcA = 1	Register A is the first ALU input.
	B	ALUSrcB = 00	Register B is the second ALU input.
SRC2	4	ALUSrcB = 01	Use 4 as the second ALU input.
	Extend	ALUSrcB = 10	Use output of the sign extension unit as the second ALU input.
	Extshft	ALUSrcB = 11	Use the output of the shift-by-two unit as the second ALU input.
	Read		Read two registers using the rs and rt fields of the IR as the register numbers and putting the data into registers A and B.
Register control	Write ALU	RegWrite, RegDst = 1, MemtoReg = 0	Write a register using the rd field of the IR as the register number and the contents of the ALUOut as the data.
	Write MDR	RegWrite, RegDst = 0, MemtoReg = 1	Write a register using the rt field of the IR as the register number and the contents of the MDR as the data.
	Read PC	MemRead, lorD = 0	Read memory using the PC as address; write result into IR (and the MDR).
Memory	Read ALU	MemRead, lorD = 1	Read memory using the ALUOut as address; write result into MDR.
	Write ALU	MemWrite, lorD = 1	Write memory using the ALUOut as address, contents of B as the data.
	ALU	PCSource = 00 PCWrite	Write the output of the ALU into the PC.
PC write control	ALUOut-cond	PCSource = 01, PCWriteCond	If the Zero output of the ALU is active, write the PC with the contents of the register ALUOut.
	jump address	PCSource = 10, PCWrite	Write the PC with the jump address from the instruction.
	Seq	AddrCtl = 11	Choose the next microinstruction sequentially.
Sequencing	Fetch	AddrCtl = 00	Go to the first microinstruction to begin a new instruction.
	Dispatch 1	AddrCtl = 01	Dispatch using the ROM 1.
	Dispatch 2	AddrCtl = 10	Dispatch using the ROM 2.