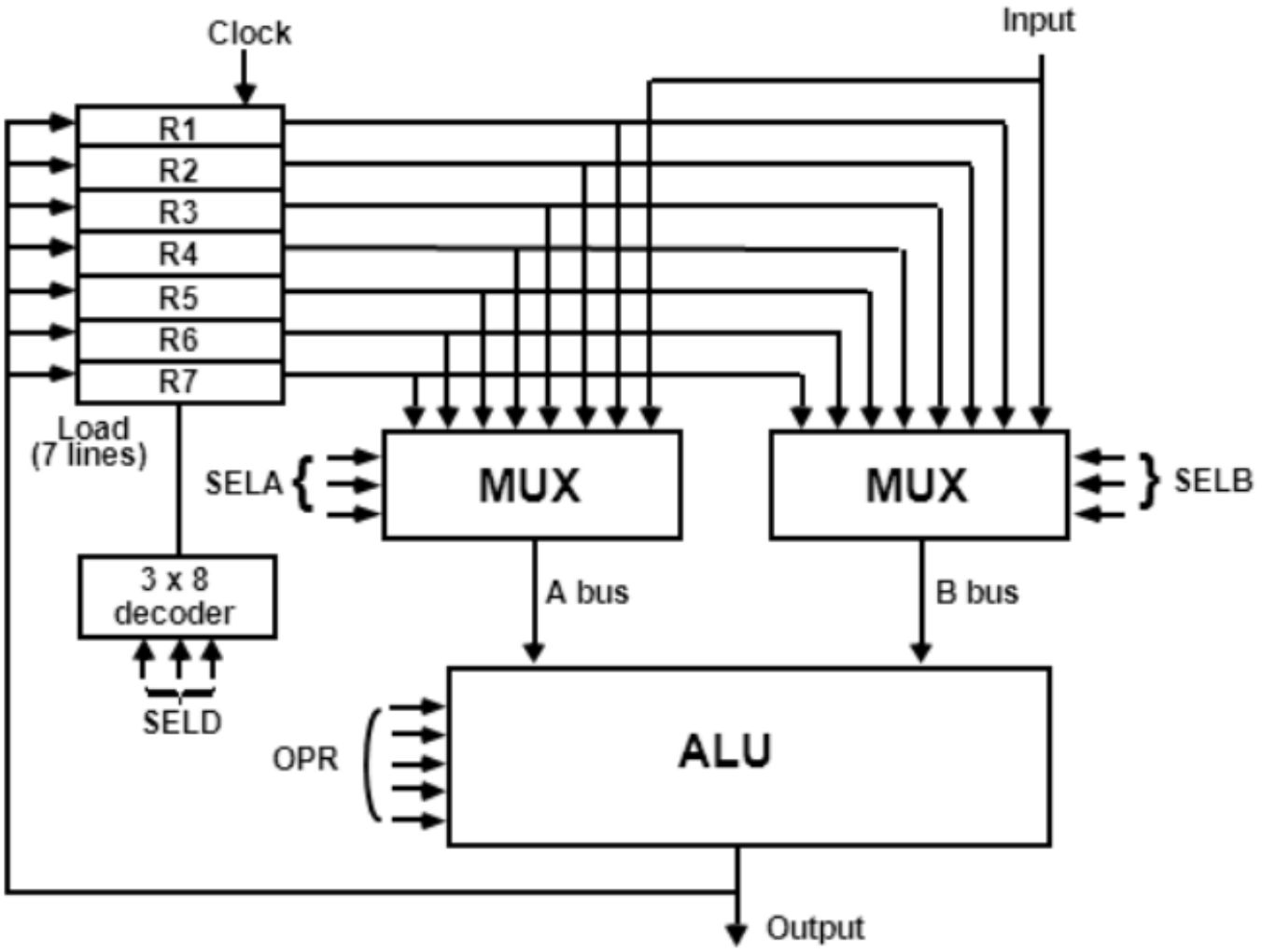
Unit 7
Processor Logic Design

BIT
Digital Logic
1st Sem

PROCESSOR ORGANIZATION

- In general, most processors are organized in one of 3 ways
  - Single register (Accumulator) organization
    - » Basic Computer is a good example
    - » Accumulator is the only general purpose register
  - General register organization
    - » Used by most modern computer processors
    - » Any of the registers can be used as the source or destination for computer operations
  - Stack organization
    - » All operations are done using the hardware stack
    - » For example, an OR instruction will pop the two top elements from the stack, do a logical OR on them, and push the result on the stack

GENERAL REGISTER ORGANIZATION



OPERATION OF CONTROL UNIT

The control unit

Directs the information flow through ALU by

- Selecting various *Components* in the system
- Selecting the *Function* of ALU

Example: R1 ← R2 + R3

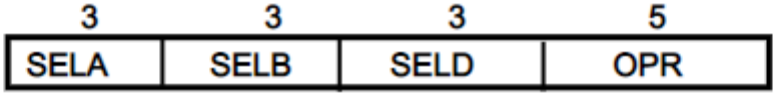
[1] MUX A selector (SELA): BUS A ← R2

[2] MUX B selector (SELB): BUS B ← R3

[3] ALU operation selector (OPR): ALU to ADD

[4] Decoder destination selector (SELD): R1 ← Out Bus

Control Word



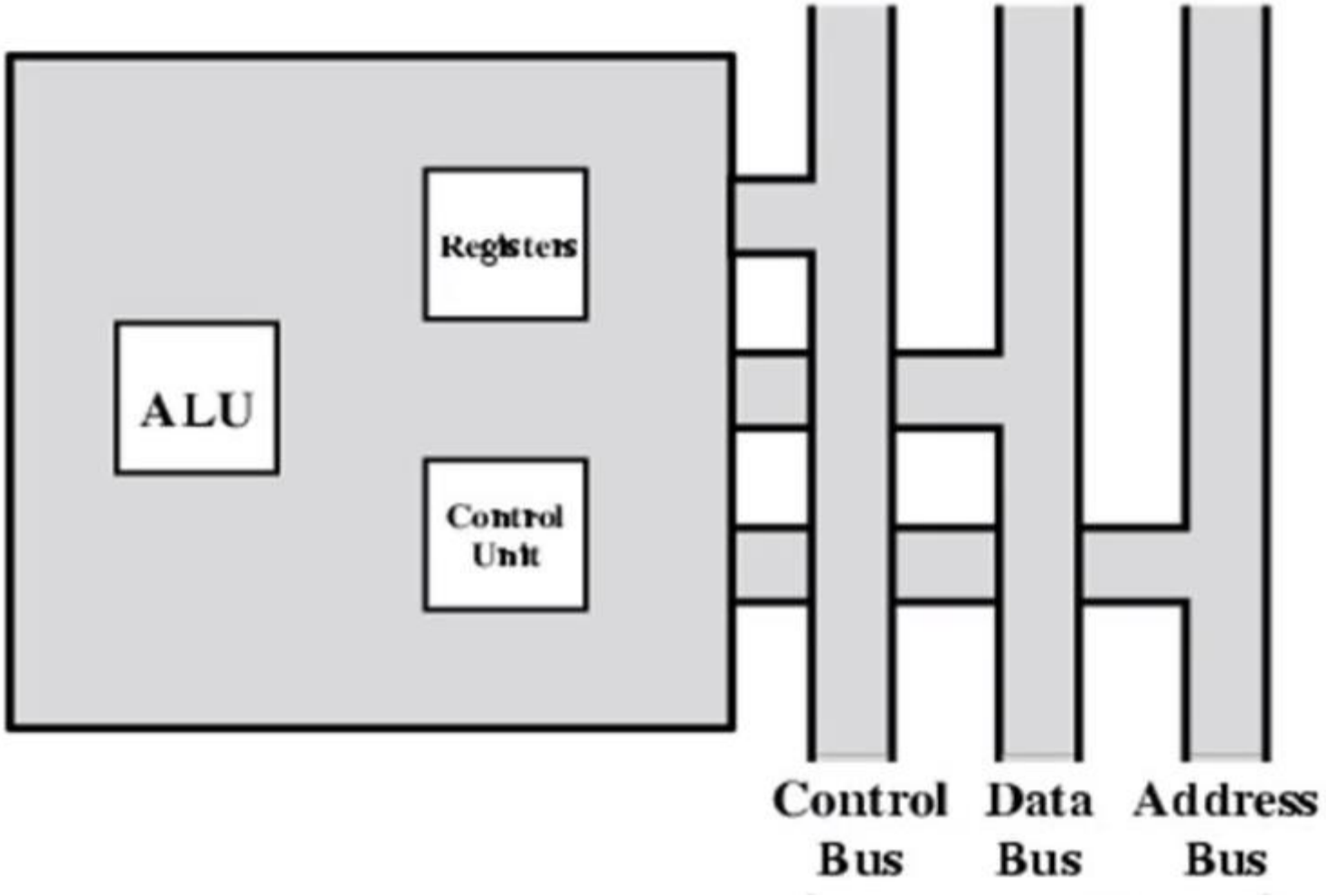
Encoding of register selection fields

Binary Code	SELA	SELB	SELD
000	Input	Input	None
001	R1	R1	R1
010	R2	R2	R2
011	R3	R3	R3
100	R4	R4	R4
101	R5	R5	R5
110	R6	R6	R6
111	R7	R7	R7

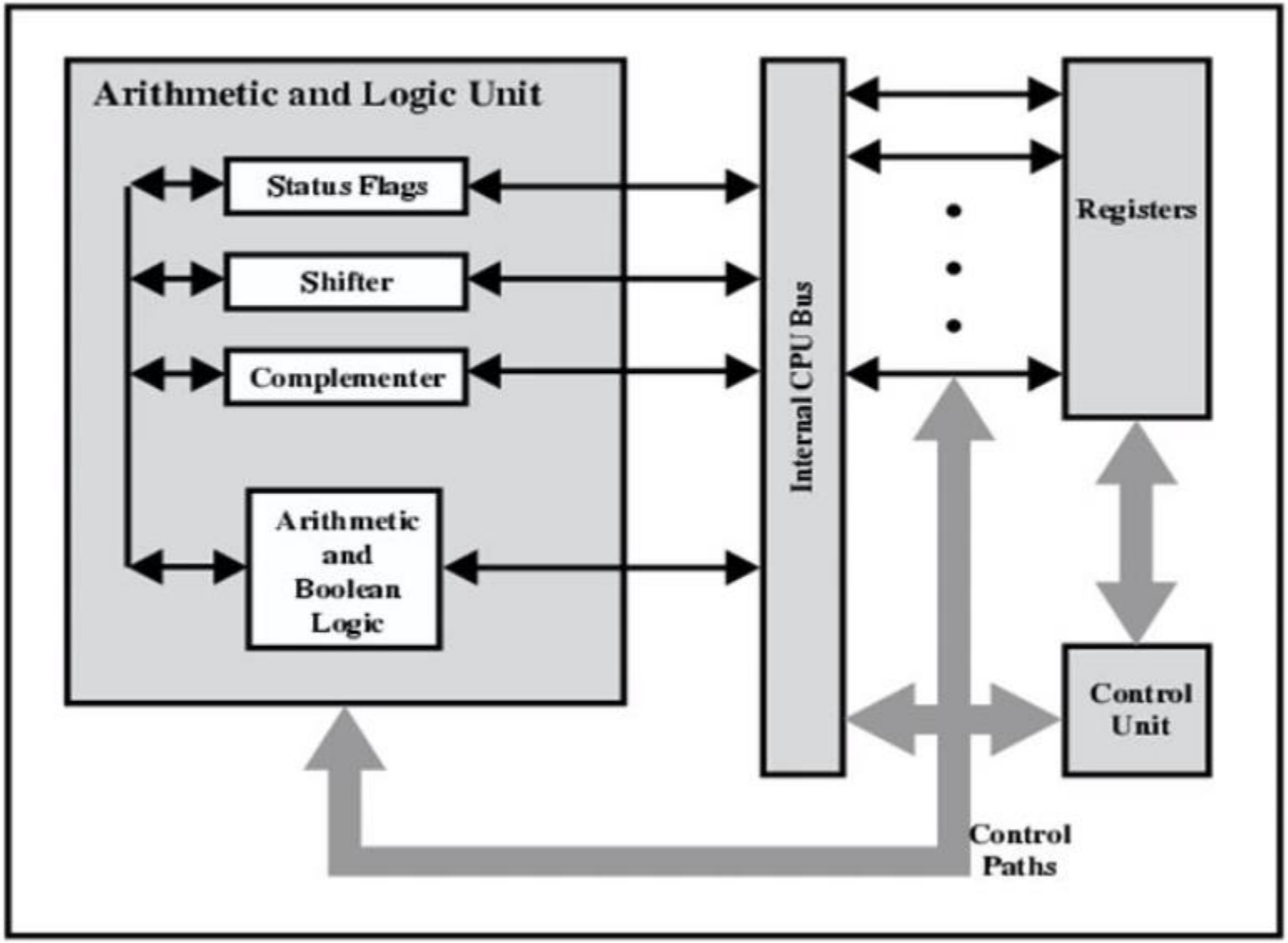
Processor Organization

- CPU must do:
  - Fetch instructions
  - Interpret instructions
  - Fetch data
  - Process data
  - Write data
- CPU must have some working space (temporary storage) called registers

CPU With Systems Bus



CPU Internal Structure



Registers

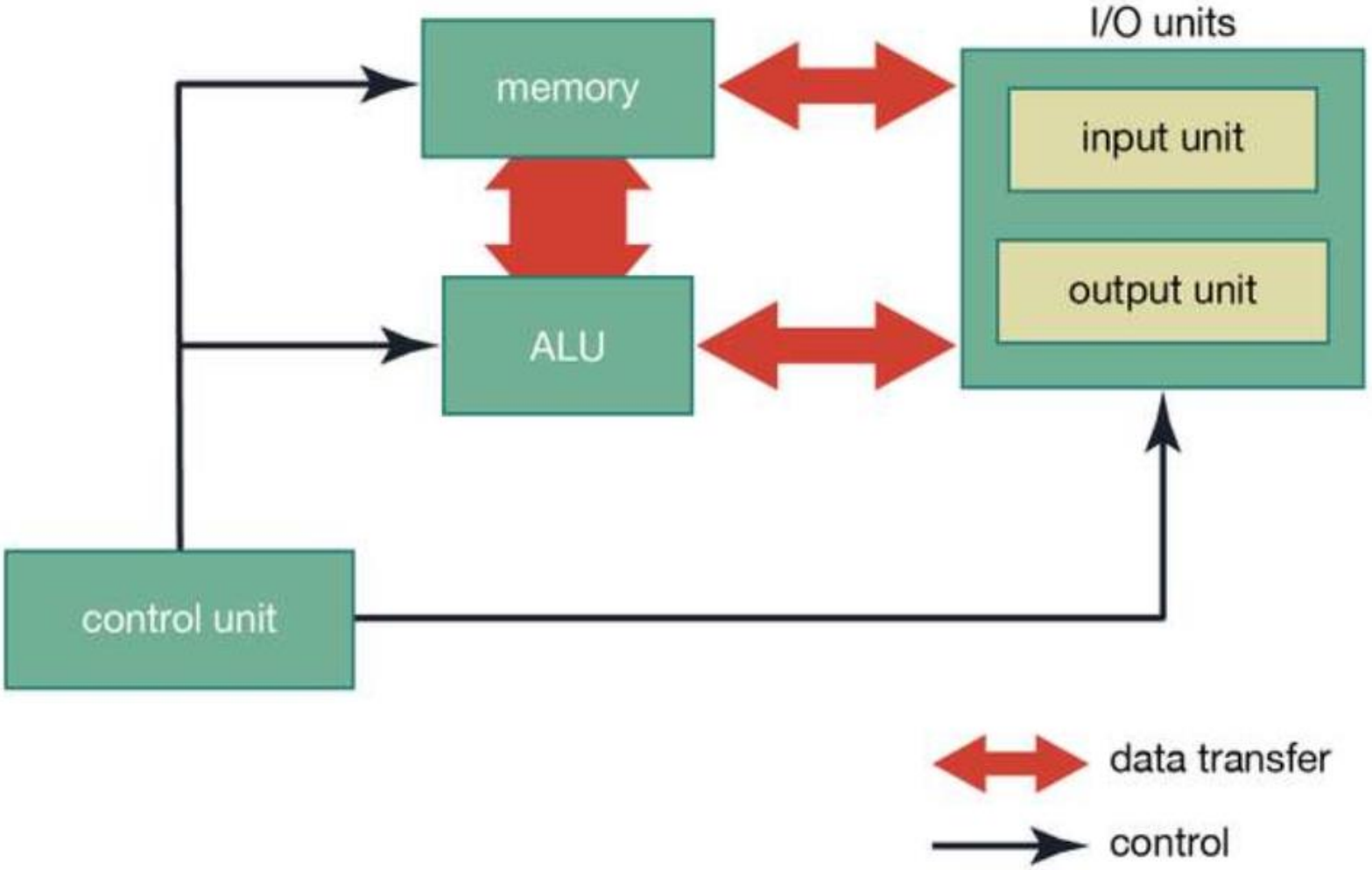
- Number and function vary between processor designs
- Top level of memory hierarchy (faster, smaller, and more expensive per bit)
  - User Visible Registers: Enable assembly language programmer to minimize main memory references.
  - Control and status registers: Used by the CU to control the operation of the processor and by operating system programs to control the execution of programs

Program Status Word (PSW)

- A set of bits
- Includes Condition Codes such as:
  - Sign
  - Zero
  - Carry
  - Equal
  - Overflow
- Interrupt enable/disable
- Supervisor

- Memory Address Register (MAR)
  - Connected to address bus
  - Specifies address for read or write operand
- Memory Buffer Register (MBR)
  - Connected to data bus
  - Holds data to write or last data read
- Program Counter (PC)
  - Holds address of next instruction to be fetched
- Instruction Register (IR)
  - Holds last instruction fetched

Arithmetic Logic Unit



Arithmetic Logic Unit

- It is the size of the word that the ALU can handle which, more than any other measure, determines the word-size of a processor: that is, a 3Z-bit processor is one with a 32-bit ALU.
- The simplest sort of ALU performs only addition, BOOLEAN LOGIC (including the NOT or complement operation) and shifts a word one bit to the right or left, all other arithmetic operations being synthesized from sequences of these primitive operations.
- For example, subtraction is performed as complement-add, multiplication by a power of two by shifting, division by repeated subtraction.

Arithmetic Circuit

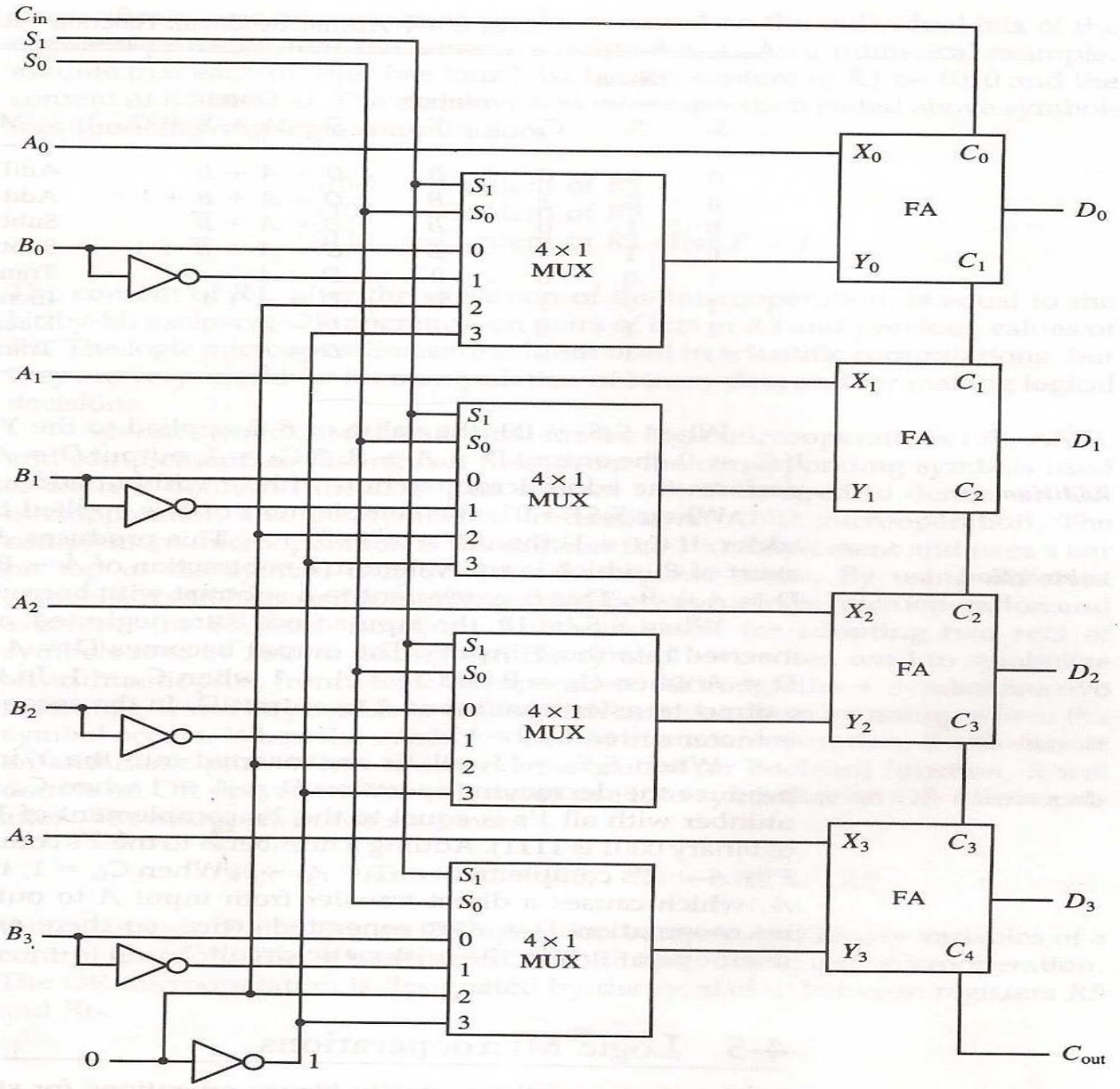
- The basic arithmetic microoperations are
  - Addition
  - Subtraction
  - Increment
  - Decrement
- The additional arithmetic microoperations are
  - Add with carry
  - Subtract with borrow
  - Transfer/Load

Arithmetic Circuit

Summary of typical arithmetic microoperations

Symbolic Designation	**Description**
R3 ← R1 + R2	Contents of R1 plus R2 transferred to R3
R3 ← R1 + R2' + 1	Contents of R1 minus R2 transferred to R3
R1 ← R1 + 1	Increment the contents of R1 by one
R1 ← R1 - 1	Decrement the contents of R1 by one
R3 ← R1 + R2 + 1	Add with carry
R3 ← R1 + R2'	Subtract with borrow
R1 ← R1' +1	2's complement the contents of R1 (negate)

Arithmetic Circuit



Arithmetic Circuit

- Arithmetic circuit Operation mechanism:

➢ When S_1S_0=00, the value of B is applied to Y inputs of adder. If C_{in}=0, Output D=A+B. If C_{in}=1, Output D=A+B+1.Both cases perform add operation with or without carrying input carry.

➢ When S_1S_0=01, the complement of B is applied to Y inputs of adder. If C_{in}=0, Output D=A+B’; this is equivalent to a subtract with borrow(A-B-1).If C_{in}=1, Output D=A+B’+1; which is equivalent to a subtraction of A-B.

➢ When S_1S_0=10, the inputs from B are neglected and instead all 0’s are inserted to Y inputs. The Output becomes D=A+0+Cin.This gives D=A when C_{in}=0 and D=A+1 when C_{in}=1.In the first case there is direct transfer from input A to output D. In the second case value of A is incremented by 1.

➢ When S_1S_0=11, all 1’s are inserted into Y inputs of the adder to produce the decrement operation D=A-1 when C_{in}=0.When C_{in}=1 then D=A-1+1=A, which causes a direct transfer from input A to output D.

Arithmetic Circuit

Select			Input	Output	
S_1	S_0	C_{in}	Y	$D = A + Y + C_{in}$	Microoperation
0	0	0	B	$D = A + B$	Add
0	0	1	B	$D = A + B + 1$	Add with carry
0	1	0	$\overline{B}$	$D = A + \overline{B}$	Subtract with borrow
0	1	1	$\overline{B}$	$D = A + \overline{B} + 1$	Subtract
1	0	0	0	$D = A$	Transfer A
1	0	1	0	$D = A + 1$	Increment A
1	1	0	1	$D = A - 1$	Decrement A
1	1	1	1	$D = A$	Transfer A

Design of Logic Circuit

Logic microoperations

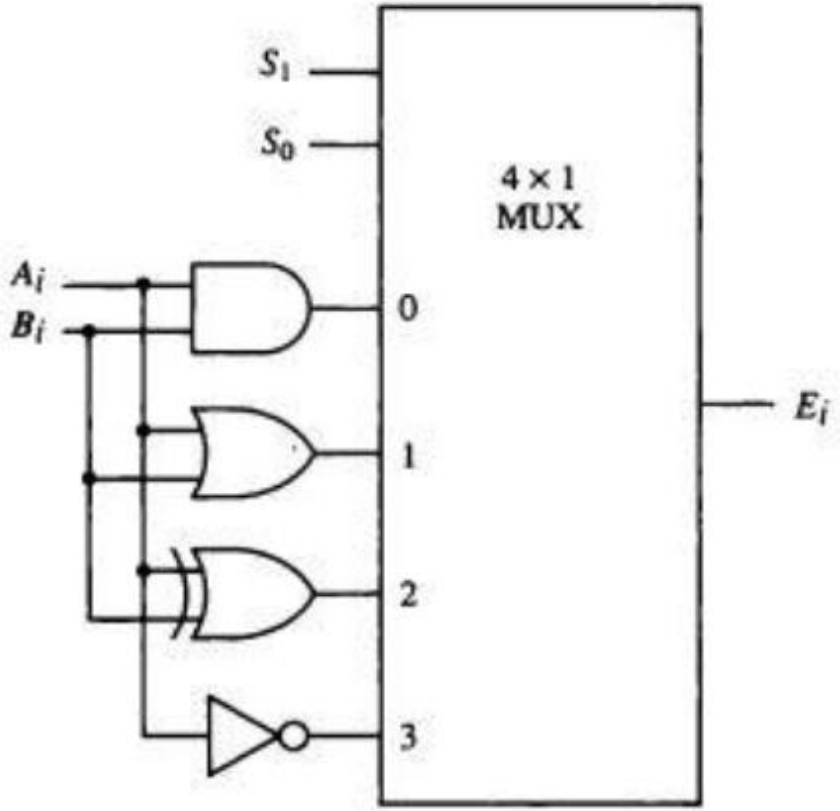
Logic microoperations are bit-wise operations, i.e., they work on the individual bits of data. Useful for bit manipulations on binary data and for making logical decisions based on the bit value. There are, in principle, 16 different logic functions that can be defined over two binary input variables. However, most systems only implement four of these

- AND ($\wedge$), OR ($\vee$), XOR ($\oplus$), Complement/NOT

The others can be created from combination of these four functions.

Microoperation	**Name**
$F \leftarrow R1 \wedge R2'$	AND
$F \leftarrow R1 \vee R2$	OR
$F \leftarrow R1 \oplus R2$	XOR
$F \leftarrow R1'$	Complement (NOT)

Design of Logic Circuit



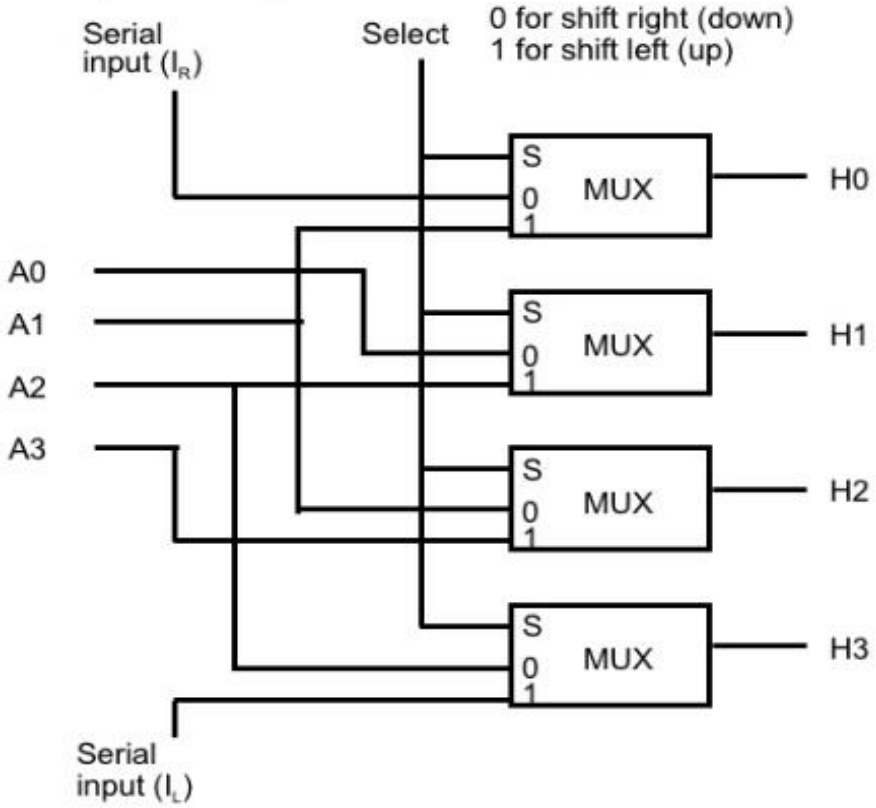
(a) Logic diagram

S_1	S_0	Output	Operation
0	0	$E = A \wedge B$	AND
0	1	$E = A \vee B$	OR
1	0	$E = A \oplus B$	XOR
1	1	$E = \overline{A}$	Complement

(b) Function table

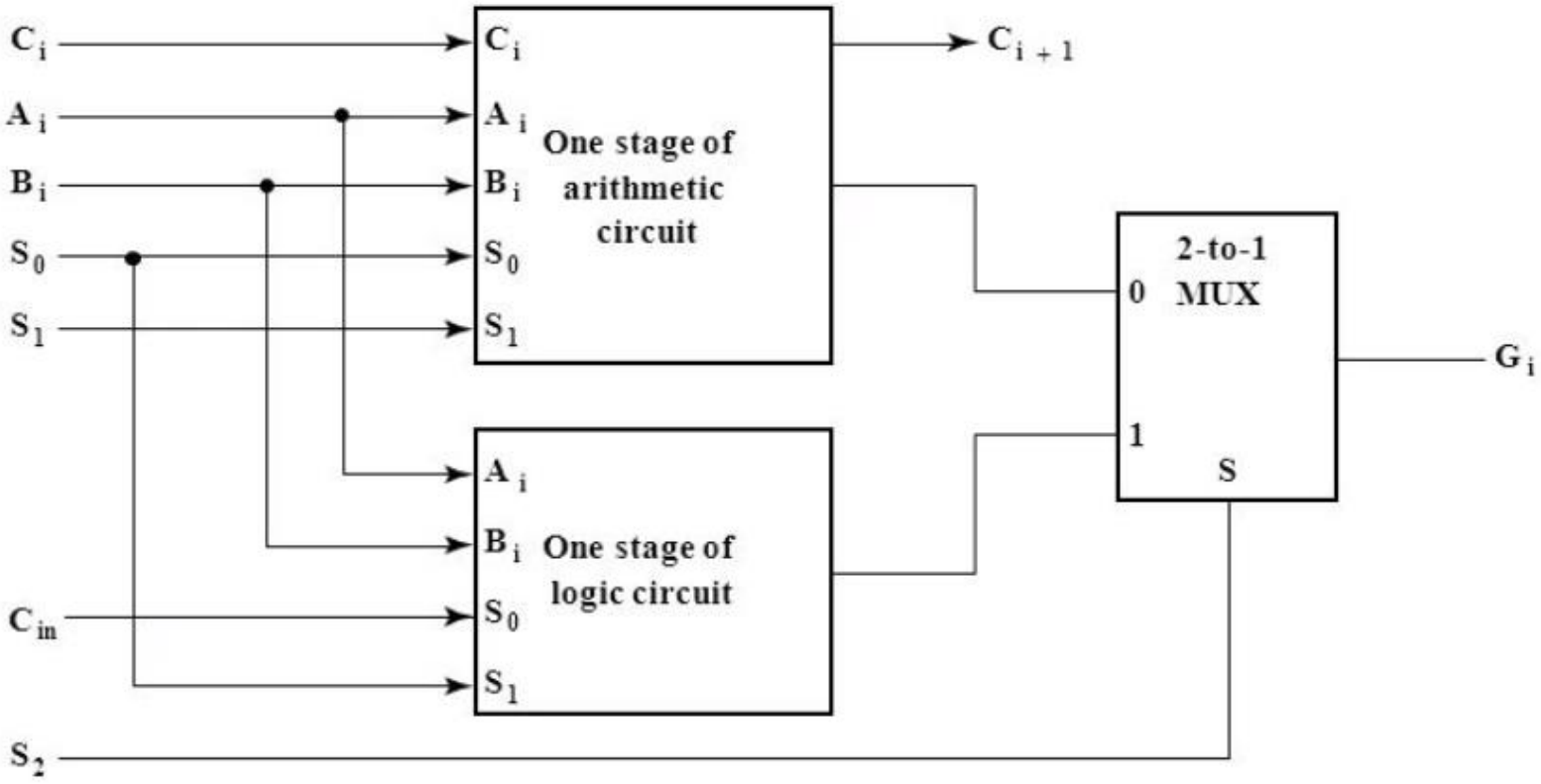
Design of Shifter

A combinational circuit shifter can be constructed with multiplexers. The 4-bit shifter has four Data inputs, A0 through A3 and four data outputs, H0 through H3. There are two serial inputs, one For shift left (IL) and the other for shift right (IR). When the selection input S=0 , the input data are Shifted right (down in the diagram). When S=1, the input data are shifted left(up in the diagram). The function table shows which input goes to each output after the shift. A shifter with n data and Outputs requires n multiplexers. The two serial inputs can be controlled another multiplexer to Provide the three possible types of shifts.



Select	Output			
S	H0	H1	H2	H3
0	I_R	A0	A1	A2
1	A1	A2	A3	I_L

Design of Arithmetic Logic Unit



For *n*-bit ALU the one stage circuit must be repeated *n* times.

Design of Arithmetic Logic Unit

Function Table for ALU

Operation Select					
S_2	S_1	S_0	C_{in}	Operation	Function
0	0	0	0	$G \leftarrow A$	Transfer A
0	0	0	1	$G \leftarrow A + 1$	Increment A
0	0	1	0	$G \leftarrow A + B$	Addition
0	0	1	1	$G \leftarrow A + B + 1$	Add with carry input of 1
0	1	0	0	$G \leftarrow A + \overline{B}$	A plus 1's complement of B
0	1	0	1	$G \leftarrow A + \overline{B} + 1$	Subtraction
0	1	1	0	$G \leftarrow A + 1$	Decrement A
0	1	1	1	$G \leftarrow A$	Transfer A
1	X	0	0	$G \leftarrow A \wedge B$	AND
1	X	0	1	$G \leftarrow A \vee B$	OR
1	X	1	0	$G \leftarrow A \oplus B$	XOR
1	X	1	1	$G \leftarrow \overline{A}$	NOT (1's complement)

S2 = 0 → Arithmetic operations

S2 = 1 → logic operations

Status register

- Relative magnitude of two numbers may be determined by subtracting one from the other.
- Also called as flag bits or condition code bits
- Status Bits
  - C – Carry
  - S – Sign bit
  - Z – zero bit
  - V – Overflow bit

Status Register