AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH (AIUB)
Where leaders are created

# Lecture # 4 (Final)
# Processor Logic Design (2nd Part)
## Status Register and Shifter Design

**Prof. Dr. Engr. Muhibul Haque Bhuyan**
Professor
Department of Electrical and Electronic Engineering
American International University Bangladesh (AIUB)

# Design of Status Register

- It is sometimes convenient to supplement the ALU with a status register where the **status bit** (overflow, zero indication, sign) conditions are stored for further analysis.
- Status-bit conditions are sometimes called condition-code bits or flag bits.

# Setting Bits in a Status Register

8-bit ALU with a 4-bit status register

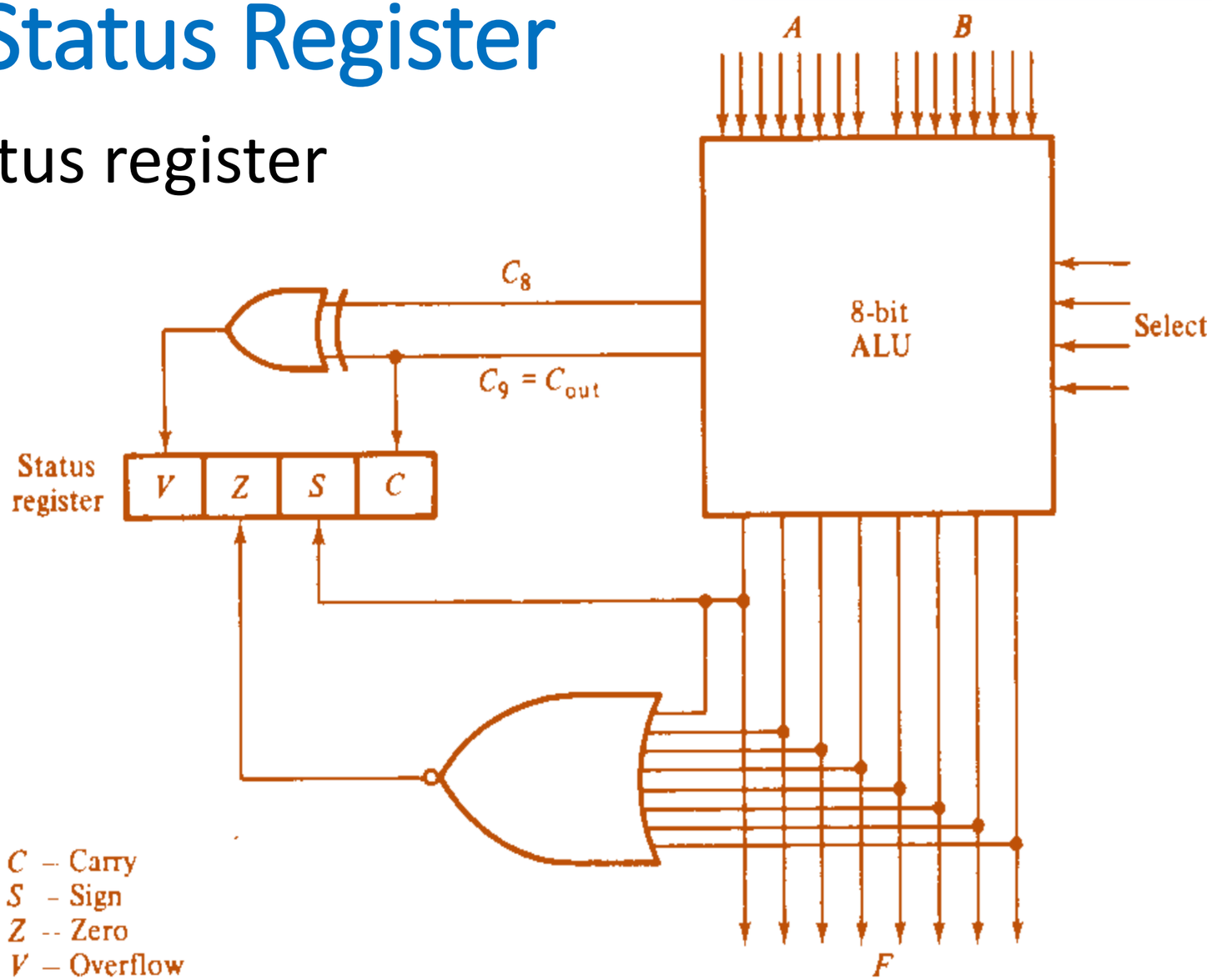



**Fig. 9-14**

# Setting Bits in a Status Register

Figure 9.14. shows the block diagram of an 8-bit ALU with a 4-bit status register. The four status bits for **carry, sign, zero**, and **overflow** bits are symbolized by *C*, *S*, *Z*, and *V* respectively. The bits are **set** (**HIGH**) or **cleared** (**LOW**) as a result of an operation performed in the ALU.

1. Bit *C* is set if the output carry of the ALU is 1. It is cleared if the output carry is 0.
2. Bit *S* is set if the highest-order bit of the result (i.e., the sign bit) in the output of the ALU is 1. It is cleared if the highest-order bit of the result is 0.
3. Bit *Z* is set if the output carry of the ALU contains all 0s, and it is cleared otherwise. That is, *Z* = 1 if the result is 0, and *Z* = 0 if the result is not 0.
4. Bit *V* is set if the exclusive-OR of output carry and MSB of the ALU is 1 (that is, $C_9 \oplus C_8 = 1$), and it is cleared otherwise. This is the condition of overflow when the numbers are in sign 2's complement representation. For an 8-bit ALU, *V* = 1 if the result is greater than 127 or less than -128.

# Table 9-5 Status bits after the subtraction operation of two **unsigned** numbers (*A*-*B*)

| Relation | Condition of status bits | Boolean function |
|---|---|---|
| $A > B$ | $C = 1$ and $Z = 0$ | $CZ'$ |
| $A \geqslant B$ | $C = 1$ | $C$ |
| $A < B$ | $C = 0$ | $C'$ |
| $A \leqslant B$ | $C = 0$ or $Z = 1$ | $C' + Z$ |
| $A = B$ | $Z = 1$ | $Z$ |
| $A \neq B$ | $Z = 0$ | $Z'$ |

# Status bits after the subtraction (*A*-*B*) of two signed binary numbers when negative numbers are in 2's complement form

| Relation | Condition of status bits | Boolean function |
|---|---|---|
| $A > B$ | $Z = 0$ and ($S = 0, V = 0$ or $S = 1, V = 1$) | $Z'(S \odot V)$ |
| $A \geq B$ | $S = 0, V = 0$ or $S = 1, V = 1$ | $S \odot V$ |
| $A < B$ | $S = 1, V = 0$ or $S = 0, V = 1$ | $S \oplus V$ |
| $A \leq B$ | $S = 1, V = 0$ or $S = 0, V = 1$ or $Z = 1$ | $(S \oplus V) + Z$ |
| $A = B$ | $Z = 1$ | $Z$ |
| $A \neq B$ | $Z = 0$ | $Z'$ |

# Status bits after the subtraction (*A*-*B*) of two **signed** binary numbers when negative numbers are in 2's complement form

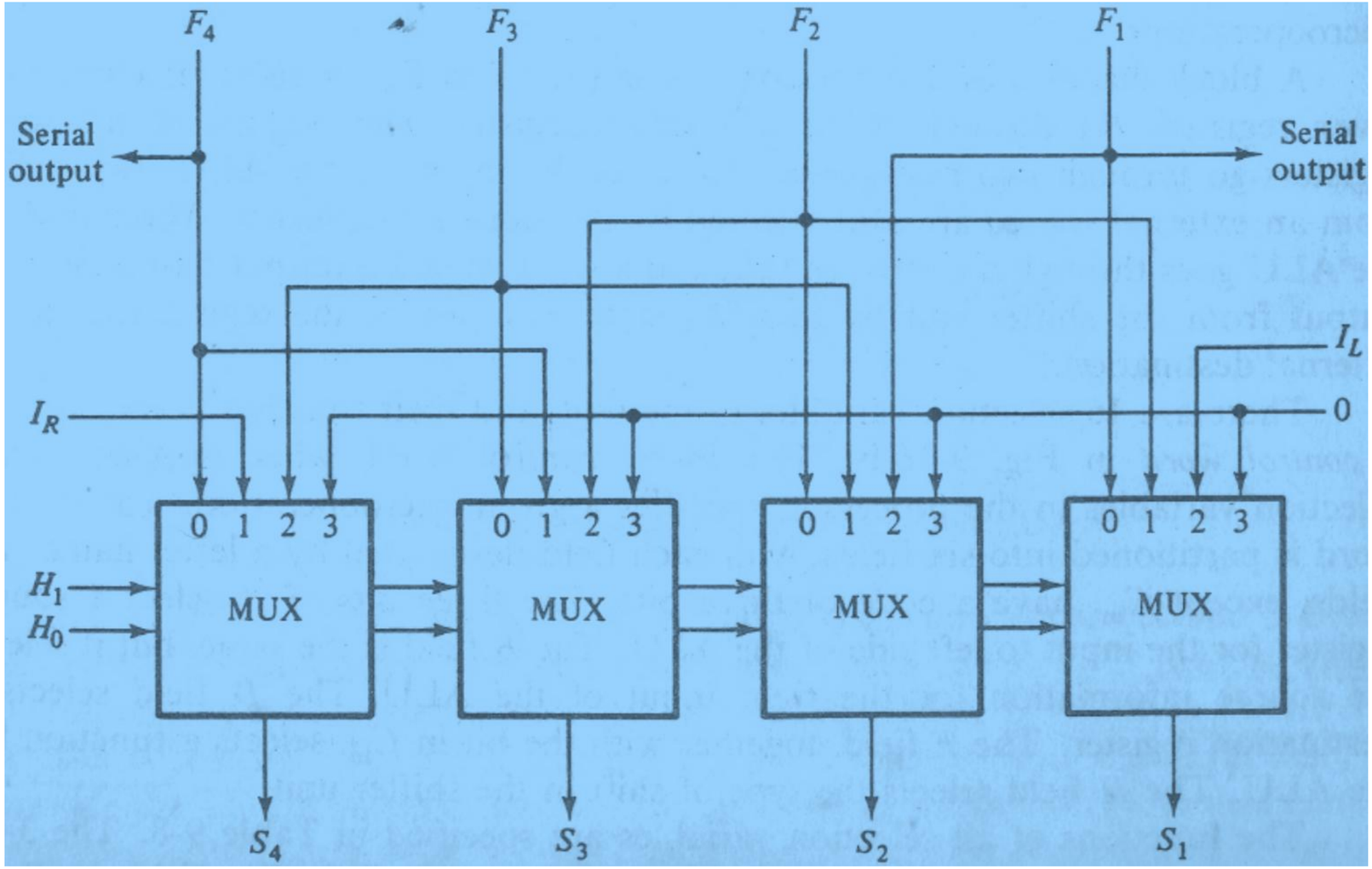
Some computers consider the ***C* bit to be a borrow bit** after a subtraction operation *A* – *B*. An end **borrow does not occur if *A* ≥ *B***, but **an extra bit must be borrowed when *A* < *B***.

The **condition for a borrow** is the **complement of the output carry** obtained when the subtraction is done by taking the 2's complement of B. For this reason, a processor that considers the *C* bit to be a borrow after a subtraction will complement the *C* bit after a subtraction or compare operation and denote this bit as a borrow.

# Design of a Shifter

- The shift unit attached to a processor transfers the output of the ALU onto the output bus.
- The shifter may transfer the information directly without a shift, or it may shift the information to the right or left.
- Provision is sometimes made for no transfer from the ALU to the output bus.
- The shifter provides the shift micro-operations commonly not available in an ALU.

# Design of Shifter



# Function Table for Shifter

| $H_1$ $H_0$ | Operation | Function |
|---|---|---|
| 0 0 | $S \leftarrow F$ | Transfer *F* into the output Bus, *S* (No Shift) |
| 0 1 | $S \leftarrow$ shr *F* | Shift Right *F* into the output Bus, *S* |
| 1 0 | $S \leftarrow$ shl *F* | Shift Left *F* into the output Bus, *S* |
| 1 1 | $S \leftarrow 0$ | Transfer 0's into the output Bus, *S* |

To add more operations in the shifter 8X1 MUX are needed with a third selection variable $\mathbf{H_2}$. If CLC L or CRC R is used, $I_L$ and $I_R$ must be connected to Carry (*C*) respectively.

CLC L – Circulate Left with Carry

CRC R – Circulate Right with Carry

# SHIFTER DESIGN

Design a 3-bit shifter for the shifting operations listed in the following Table:

| Binary Code | | | Function of selection variables | | | |
|---|---|---|---|---|---|---|
| | B | A | F with $C_{in}$ = 0 | F with $C_{in}$ = 1 | D | H |
| 0 0 0 | Input Data | Input Data | A | A+1 | None | Circulate-Left with Carry |
| 0 0 1 | R1 | R1 | A+B | A+B+1 | R1 | Circulate-Right with Carry |
| 0 1 0 | R2 | R2 | A+B’ | A+B’+1 | R2 | No shift |
| 0 1 1 | R3 | R3 | A-1 | A | R3 | 0’s to the output Bus |
| 1 0 0 | R4 | R4 | A OR B | A OR B | R4 | - |
| 1 0 1 | R5 | R5 | A XOR B | A XOR B | R5 | Shift Left with $I_L$=0 |
| 1 1 0 | R6 | R6 | A AND B | A AND B | R6 | Shift Right with $I_R$=0 |
| 1 1 1 | R7 | R7 | A’ | A’ | R7 | 1’s to the output Bus |

# SHIFTER DESIGN

Left and right serial inputs may or may not be 0 always. The serial inputs may be given as per requirements.

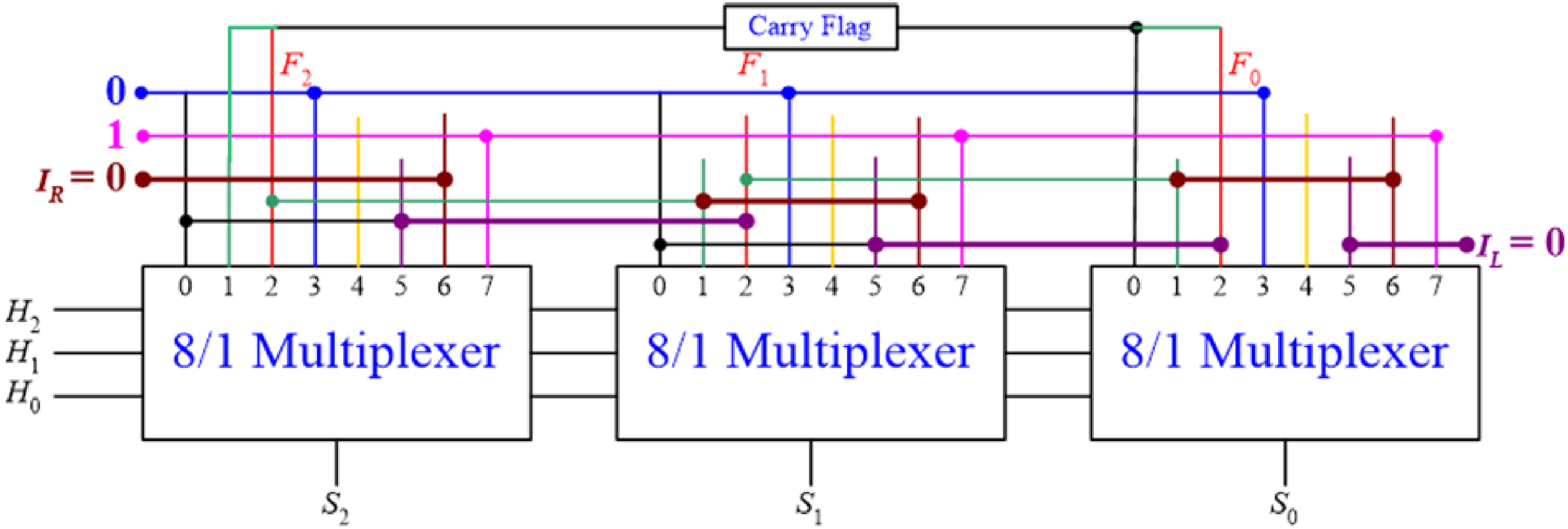



Fig. 3-bit shifter circuit for 7 different operations using three 8:1 multiplexers

# Design of a Processor Unit

The selection variables in a processor unit control the micro-operations executed within the processor during any given clock pulse. The selection variables control the buses, the ALU, the shifter, and the destination register. We will now demonstrate how the control variables select the micro-operations in a processor unit. The examples define a processor unit together with all selection variables. We will also discuss the choice of control variables for some typical micro-operations.

The block of a processor unit is shown in Fig. 9.10 (a). It consists of seven registers (R1 to R7) and a status register. The outputs of the seven registers go through two multiplexers to select the inputs to the ALU.

Input data from an external source also selected by the same multiplexers.

The output of the ALU goes through a shifter and then to a set of external output terminals. The output from the shifter can be transferred to any one of the registers or to an external destination.

# Design of a Processor Unit

There are 16 selection variables in the unit, and their function is specified by a control word in Fig. 9.16 (b). **The 16-bit control word**, when applied to the selection variables in the processor, **specifies a given micro-operation**.

The control word is partitioned into six fields, with each field designated by a letter name. **All fields, except $C_{in}$, have a code of three bits**.

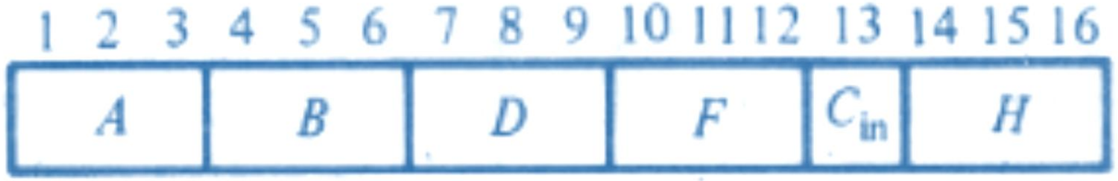



Fig. 9.16 (b) Control Word Format

The three bits of *A* field select a source register for the input to left side of the ALU.
The three bits of *B* field select a source register for the input to right side of the ALU.
The three bits of *D* field select a destination register for the output.
The three bits of *F* field together with the bit in $C_{in}$ select a function for the ALU.
The three bits of *H* field select a type of shift operation for the shifter unit.

# Design of a Processor Unit

The functions of all selection variables are specified in Table 9.8. The 3-bit binary code listed in the table specifies the code for each of the five fields *A*, *B*, *D*, *F*, and *H*. The register selected by *A*, *B* and *D* is one whose decimal number is equivalent to the binary number in the code.

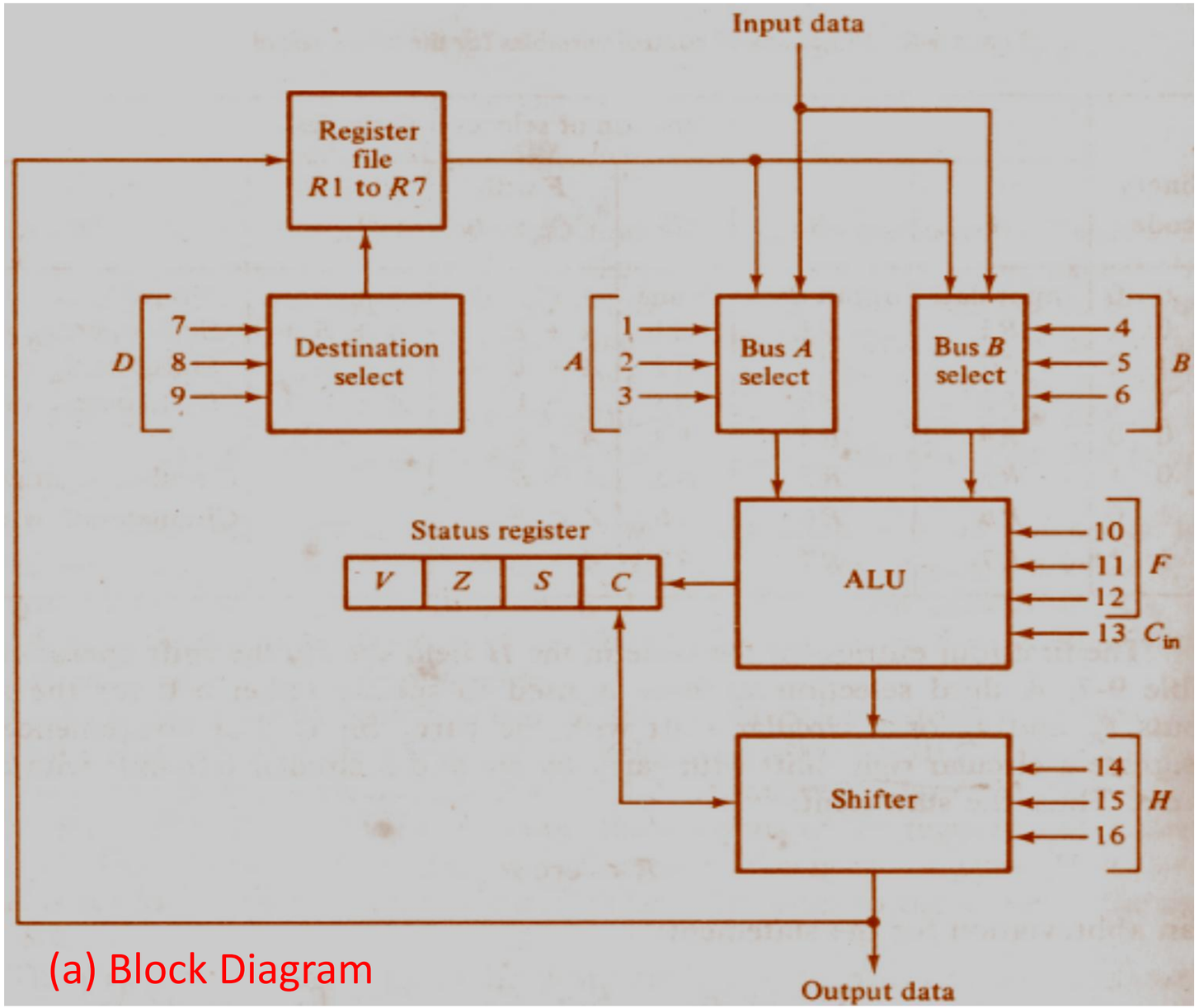
When *A* or *B* field is 000, the corresponding multiplexer selects the input data.

When *D* = 000, no destination register is selected.

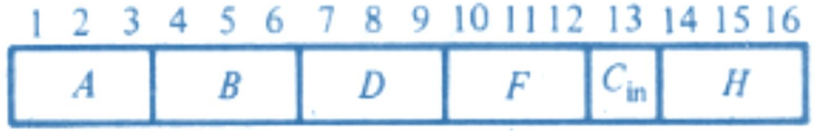
The three bits of *F* field together with the input carry bit $C_{in}$ select one of the 12 functions to be performed by the ALU as specified in Table 9.4.

Note that, there are two possibilities of transfer operation (i.e., *F* = *A*), in one case the carry bit is cleared and in the other case, the carry bit is set to high.

# Processor Unit with Control Variable



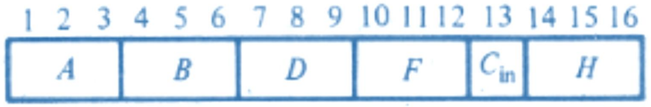

(a) Block Diagram




(b) Control word

Fig. 9-16 Processor Unit with Control Variables

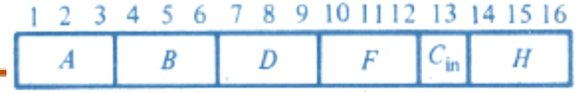
**TABLE 9-8** Functions of control variables for the processor of Fig. 9-16




(b) Control word

| Binary code | Function of selection variables | | | | | |
|---|---|---|---|---|---|---|
| | $A$ | $B$ | $D$ | $F$ with $C_{in} = 0$ | $F$ with $C_{in} = 1$ | $H$ |
| 0 0 0 | Input data | Input data | None | $A, C \leftarrow 0$ | $A + 1$ | No shift |
| 0 0 1 | $R1$ | $R1$ | $R1$ | $A + B$ | $A + B + 1$ | Shift-right, $I_R = 0$ |
| 0 1 0 | $R2$ | $R2$ | $R2$ | $A - B - 1$ | $A - B$ | Shift-left, $I_L = 0$ |
| 0 1 1 | $R3$ | $R3$ | $R3$ | $A - 1$ | $A, C \leftarrow 1$ | 0's to output bus |
| 1 0 0 | $R4$ | $R4$ | $R4$ | $A \vee B$ | — | — |
| 1 0 1 | $R5$ | $R5$ | $R5$ | $A \oplus B$ | — | Circulate-right with $C$ |
| 1 1 0 | $R6$ | $R6$ | $R6$ | $A \wedge B$ | — | Circulate-left with $C$ |
| 1 1 1 | $R7$ | $R7$ | $R7$ | $\bar{A}$ | — | — |

## TABLE 9-4 Function table for the ALU of Fig. 9-13




(b) Control word

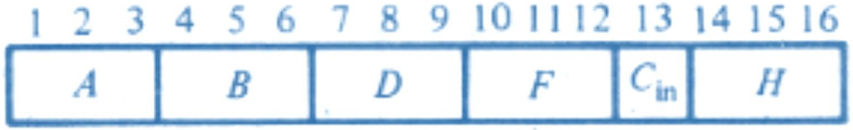
| Selection | | | | Output | Function |
|---|---|---|---|---|---|
| $s_2$ | $s_1$ | $s_0$ | $C_{in}$ | | |
| 0 | 0 | 0 | 0 | $F = A$ | Transfer $A$ |
| 0 | 0 | 0 | 1 | $F = A + 1$ | Increment $A$ |
| 0 | 0 | 1 | 0 | $F = A + B$ | Addition |
| 0 | 0 | 1 | 1 | $F = A + B + 1$ | Add with carry |
| 0 | 1 | 0 | 0 | $F = A - B - 1$ | Subtract with borrow |
| 0 | 1 | 0 | 1 | $F = A - B$ | Subtraction |
| 0 | 1 | 1 | 0 | $F = A - 1$ | Decrement $A$ |
| 0 | 1 | 1 | 1 | $F = A$ | Transfer $A$ |
| 1 | 0 | 0 | $X$ | $F = A \vee B$ | OR |
| 1 | 0 | 1 | $X$ | $F = A \oplus B$ | XOR |
| 1 | 1 | 0 | $X$ | $F = A \wedge B$ | AND |
| 1 | 1 | 1 | $X$ | $F = \overline{A}$ | Complement $A$ |

# Logic Diagram of an Arithmetic Circuit

Table 9-2 Effect of the output carry in the arithmetic circuit of Fig. 9-8

| Function select | | | Arithmetic function | $C_{out} = 1$ if | Comments |
|---|---|---|---|---|---|
| $s_1$ | $s_0$ | $C_{in}$ | | | |
| 0 | 0 | 0 | $F = A$ | | $C_{out}$ is always 0 |
| 0 | 0 | 1 | $F = A + 1$ | $A = 2^n - 1$ | $C_{out} = 1$ and $F = 0$ if $A = 2^n - 1$ |
| 0 | 1 | 0 | $F = A + B$ | $(A + B) \geq 2^n$ | Overflow occurs if $C_{out} = 1$ |
| 0 | 1 | 1 | $F = A + B + 1$ | $(A + B) \geq (2^n - 1)$ | Overflow occurs if $C_{out} = 1$ |
| 1 | 0 | 0 | $F = A - B - 1$ | $A > B$ | If $C_{out} = 0$, then $A < B$ and $F =$ 1's complement of $(B - A)$ |
| 1 | 0 | 1 | $F = A - B$ | $A \geq B$ | If $C_{out} = 0$, then $A < B$ and $F =$ 2's complement of $(B - A)$ |
| 1 | 1 | 0 | $F = A - 1$ | $A \neq 0$ | $C_{out} = 1$, except when $A = 0$ |
| 1 | 1 | 1 | $F = A$ | | $C_{out}$ is always 1 |

# Examples of Microoperations




(b) Control word

**TABLE 9-9** Examples of microoperations for processor

| Microoperation | Control word $A$ | $B$ | $D$ | $F$ | $C_{in}$ | $H$ | Function |
|---|---|---|---|---|---|---|---|
| $R1 \leftarrow R1 - R2$ | 001 | 010 | 001 | 010 | 1 | 000 | Subtract $R2$ from $R1$ |
| $R3 - R4$ | 011 | 100 | 000 | 010 | 1 | 000 | Compare $R3$ and $R4$ |
| $R5 \leftarrow R4$ | 100 | 000 | 101 | 000 | 0 | 000 | Transfer $R4$ to $R5$ |
| $R6 \leftarrow$ Input | 000 | 000 | 110 | 000 | 0 | 000 | Input data to $R6$ |
| Output $\leftarrow R7$ | 111 | 000 | 000 | 000 | 0 | 000 | Output data from $R7$ |
| $R1 \leftarrow R1, C \leftarrow 0$ | 001 | 000 | 001 | 000 | 0 | 000 | Clear carry bit $C$ |
| $R3 \leftarrow$ shl $R3$ | 011 | 011 | 011 | 100 | 0 | 010 | Shift-left $R3$ with $I_L = 0$ |
| $R1 \leftarrow$ crc $R1$ | 001 | 001 | 001 | 100 | 0 | 101 | Circulate-right $R1$ with carry |
| $R2 \leftarrow 0$ | 000 | 000 | 010 | 000 | 0 | 011 | Clear $R2$ |

If we want to place the contents of a register into the Shifter without changing the carry bit, we can use OR Logic operations with same register selected for bot  ALU input A and B.

# Examples of Micro-operations

Write a 16-bit control word for a micro-operation of adding two numbers stored in the registers R1 and R2 with carry and then storing the results in the R5 register after shifting it to the right with no external data.

**Answer:**

**Microoperation:**

**R5 ← R1 + R2 + Cin**

**Control Word Format:**

| 1 2 3 | 4 5 6 | 7 8 9 | 10 11 12 | 13 14 15 | 16 |
|---|---|---|---|---|---|
| 001 | 010 | 101 | 001 | 1 | 101 |

**In Binary form:**

**001 010 101 0011 101b**

**In Hexadecimal form:**

**2A9Dh;**

TABLE 9-8 Functions of control variables for the processor of Fig. 9-16

| Binary code | Function of selection variables: A | B | D | F with $C_{in} = 0$ | F with $C_{in} = 1$ | H |
|---|---|---|---|---|---|---|
| 0 0 | Input data | Input data | None | $A, C \leftarrow 0$ | $A + 1$ | No shift |
| 0 0 1 | R1 | R1 | R1 | $A + B$ | $A + B + 1$ | Shift-right, $I_R = 0$ |
| 0 1 0 | R2 | R2 | R2 | $A - B - 1$ | $A - B$ | Shift-left, $I_L = 0$ |
| 0 1 1 | R3 | R3 | R3 | $A - 1$ | $A, C \leftarrow 1$ | 0's to output bus |
| 1 0 0 | R4 | R4 | R4 | $A \vee B$ | — | — |
| 1 0 1 | R5 | R5 | R5 | $A \oplus B$ | — | Circulate-right with $C$ |
| 1 1 0 | R6 | R6 | R6 | $A \wedge B$ | — | Circulate-left with $C$ |
| 1 1 1 | R7 | R7 | R7 | $\bar{A}$ | — | — |

# PROCESSOR DESIGN (HOMEWORK)

Design a processor unit for the operations listed in the following Table:

| Binary Code | Functions of selection variables | | | | | |
|---|---|---|---|---|---|---|
| | B | A | F with $C_{in}$ = 0 | F with $C_{in}$ = 1 | D | H |
| 0 0 0 | Input Data | Input Data | A | A+1 | None | Circulate-Left with Carry |
| 0 0 1 | R1 | R1 | A+B | A+B+1 | R1 | Circulate-Right with Carry |
| 0 1 0 | R2 | R2 | A+B’ | A+B’+1 | R2 | No shift |
| 0 1 1 | R3 | R3 | A-1 | A | R3 | 0’s to the output Bus |
| 1 0 0 | R4 | R4 | A OR B | A OR B | R4 | - |
| 1 0 1 | R5 | R5 | A XOR B | A XOR B | R5 | Shift Left with $I_L$=0 |
| 1 1 0 | R6 | R6 | A AND B | A AND B | R6 | Shift Right with $I_R$=0 |
| 1 1 1 | R7 | R7 | A’ | A’ | R7 | 1’s to the output Bus |

# Next…

- Flowchart and State Diagram
- Microprogrammed Control Unit Design for addition/subtraction of signed numbers.

# Thanks for Attending….