# Computer Design Basics – Part A

## Instruction cycle, the Control Word

**Gloria Beraldo** (gloria.beraldo@unipd.it)
Department of Information Engineering, University of Padova

**Topics:**

- Control and microoperations
- Instruction Set Architecture
- Single-cycle computer

**Book Reference:**

- Chapter 8

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

# Datapath representation and the Control Word

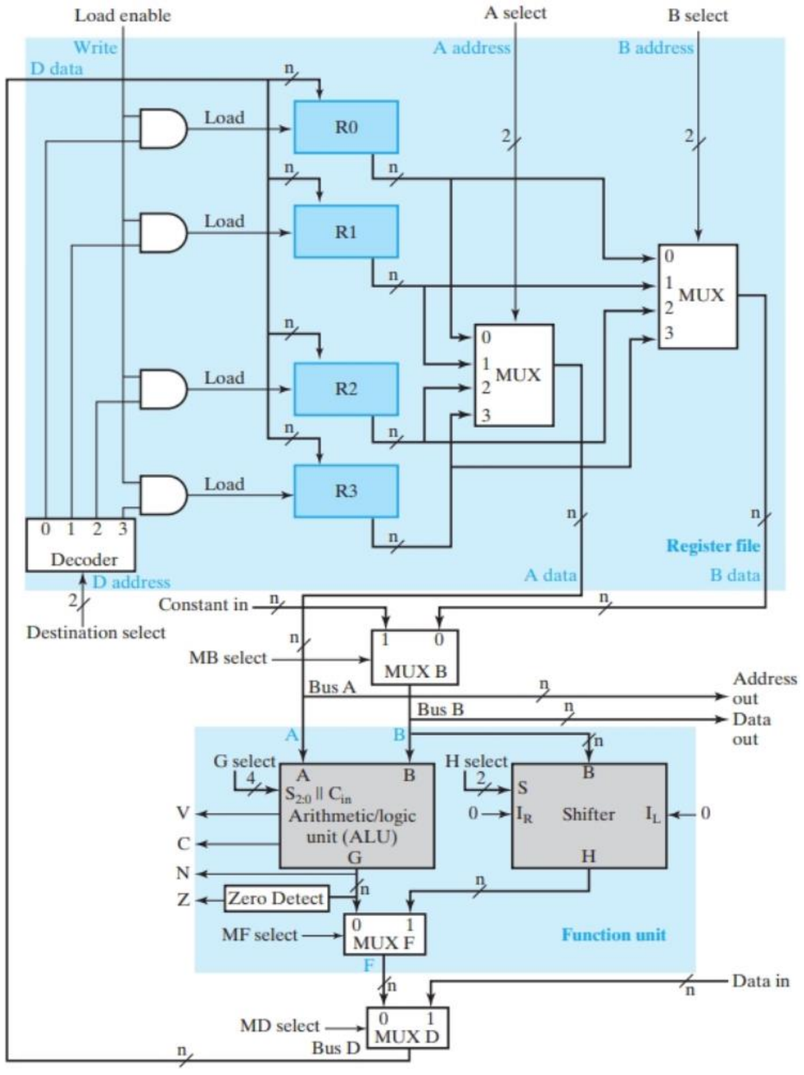



**FIGURE 8-1**
Block Diagram of a Generic Datapath

- A typical datapath has more than four registers. A set of registers may be organized into a *register file*
- The size of the register file is $2^m \times n$, where m is the number of register address bits and n is the number of bits per register
- The select of the input, operations, output is executed via the *control word* that identifies them uniquely
- The operations are executed in a single clock cycle

$2^2 = 4$ registers

# Datapaths: The Control Word (1)

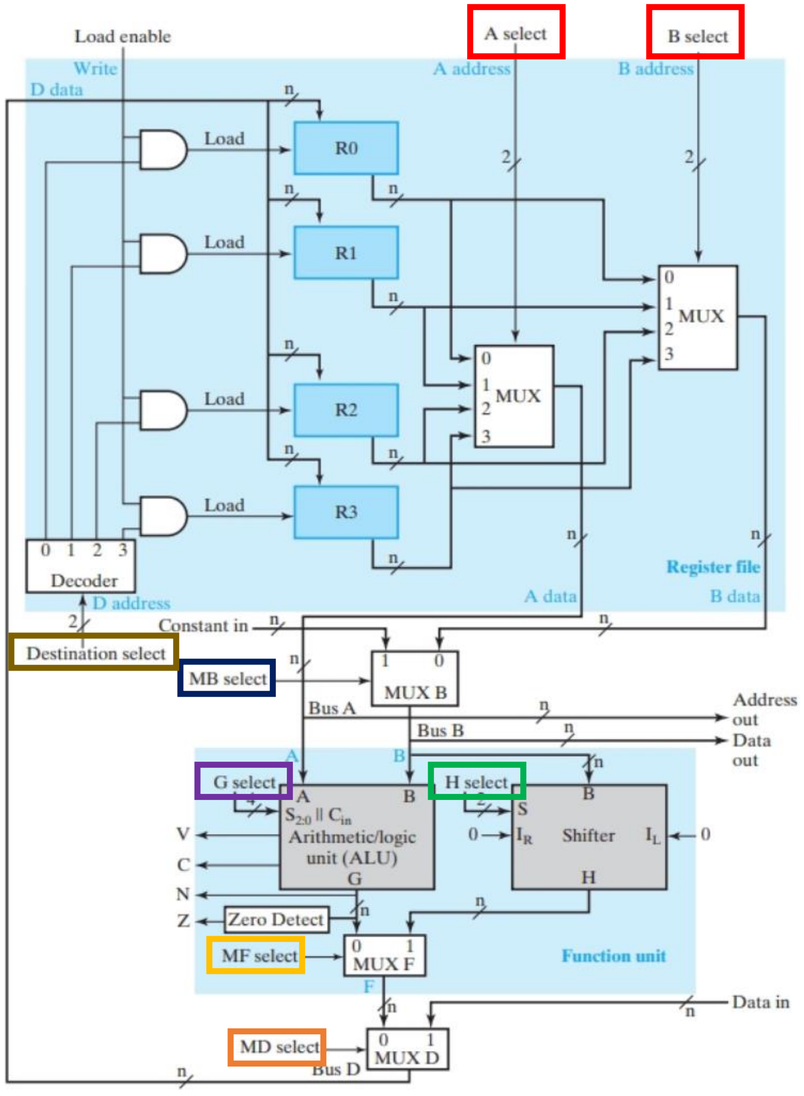



**FIGURE 8-1**
Block Diagram of a Generic Datapath

- A select, B select → select which registers to use
- G select → selects the operations that the ALU has to execute
- H select → either passes the operand on Bus B directly through to the shifter output or selects a shift microoperation
- MB select → selects whether B is achieved by a constant from outside
- MF select → selects whether the output is achieved by the ALU or by the shifter
- MD select → selects the output of the function unit or an external data
- D select → selects the register where to write the result of the operation

$2^2 = 4$ registers

# Datapaths: The Control Word (2)

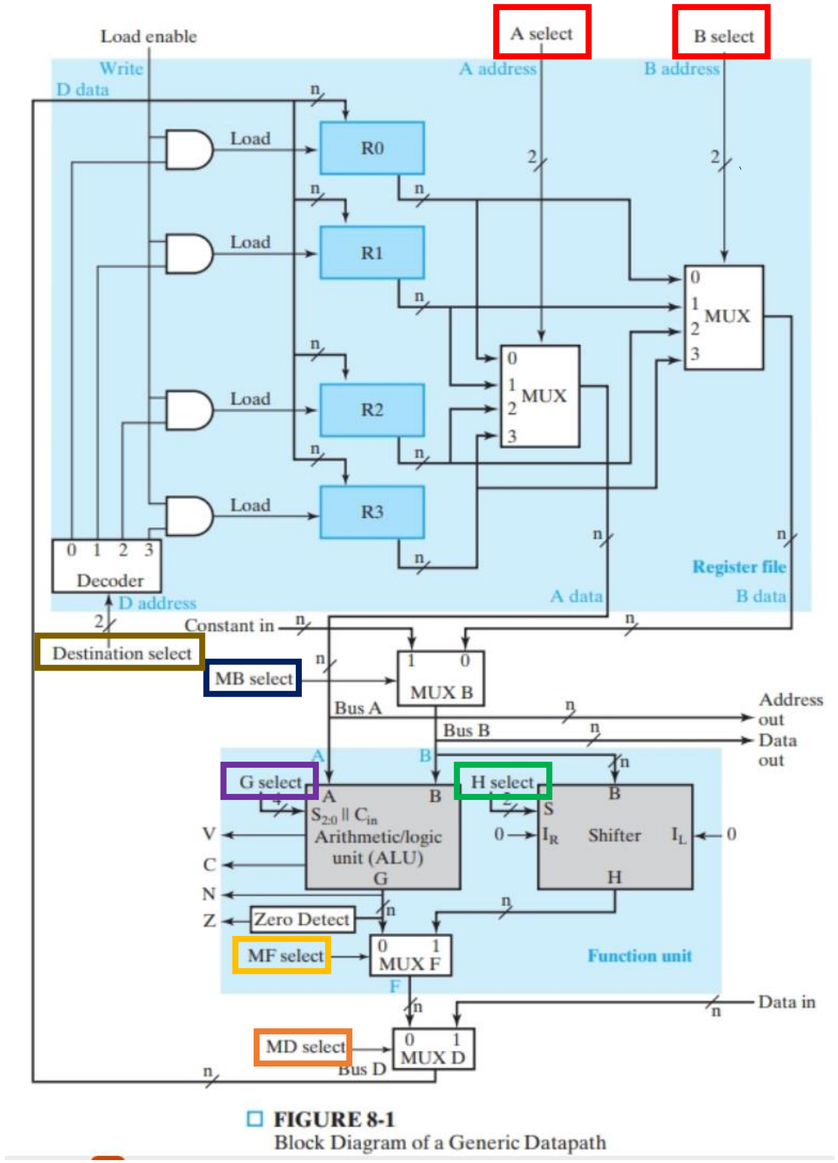
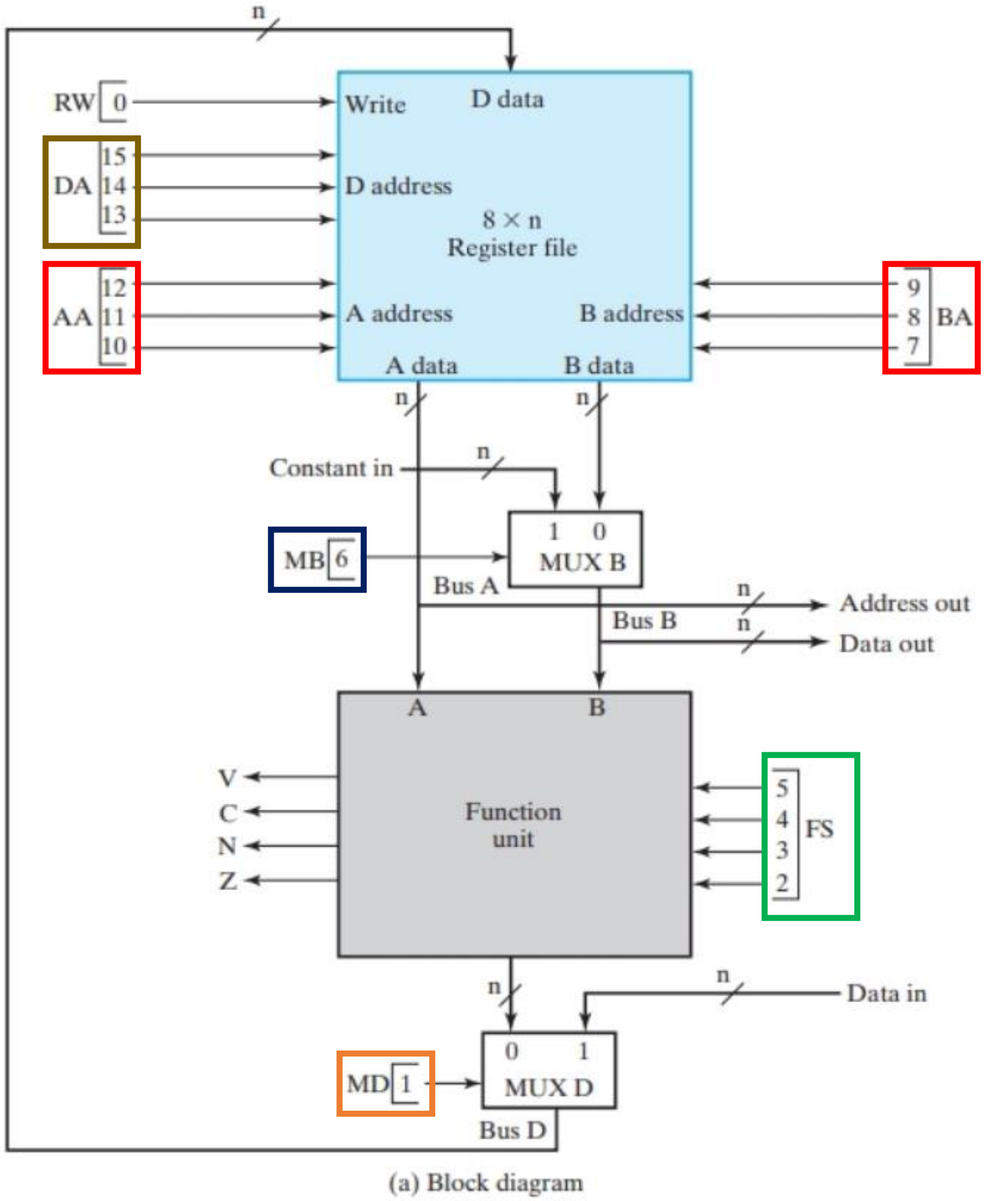



FIGURE 8-1
Block Diagram of a Generic Datapath



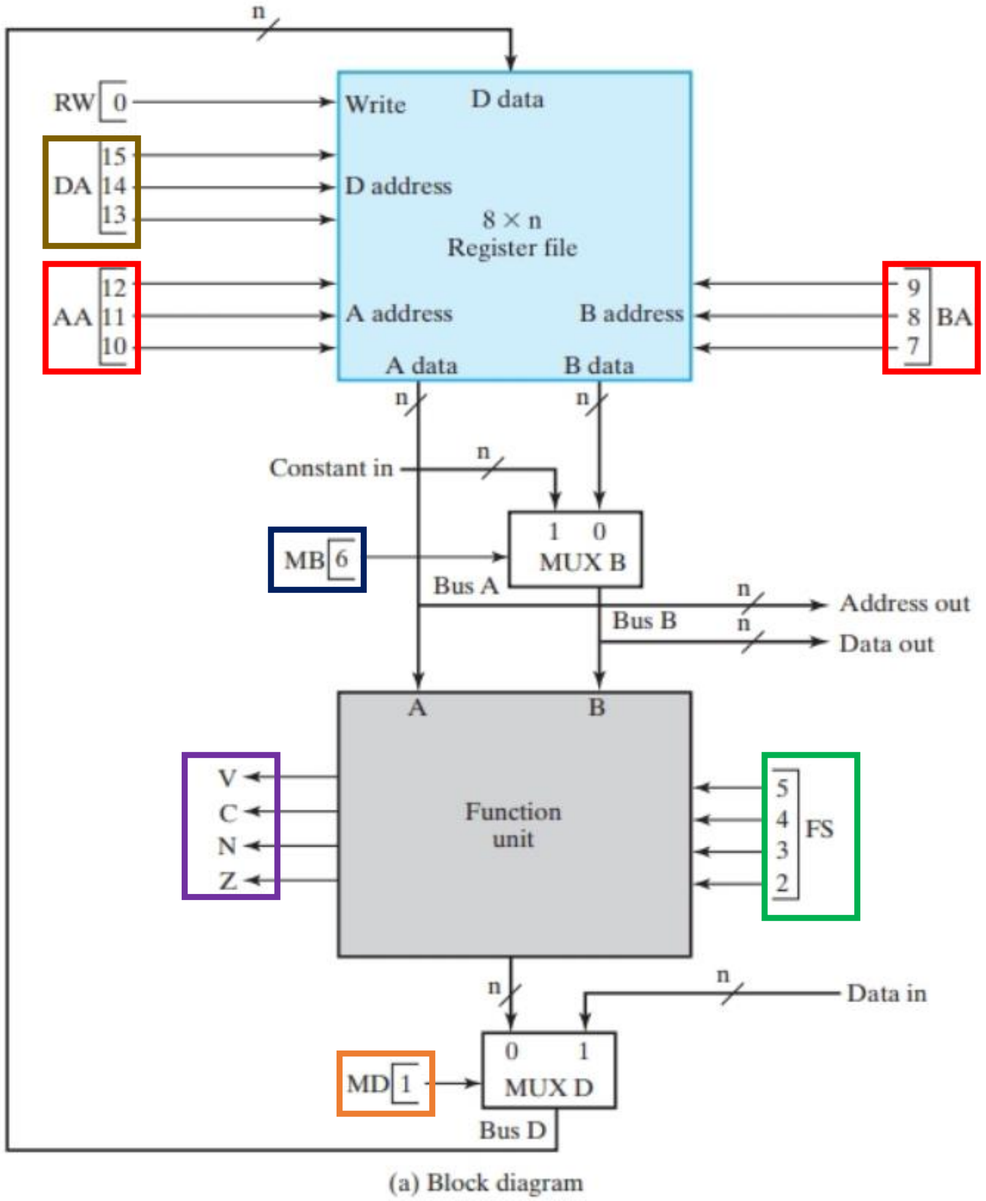

(a) Block diagram

$2^2 = 4$ registers

Now let's suppose to have $2^3 = 8$ registers

# Datapaths: The Control Word (3)

- **DA** identifies the address of the output register
- **AA** identifies the address of the register for the input A
- **BA** identifies the address of the register for the input B
- **MB** selects either the register selected by the BA field or a constant
- **FS** identifies the operation to execute (ALU or shifter)
- **MD** identifies the output
- **RW** identifies whether to write the output to the output register
- **Status bits**: V (overflow), C (carry), N (negative), Z (zero)

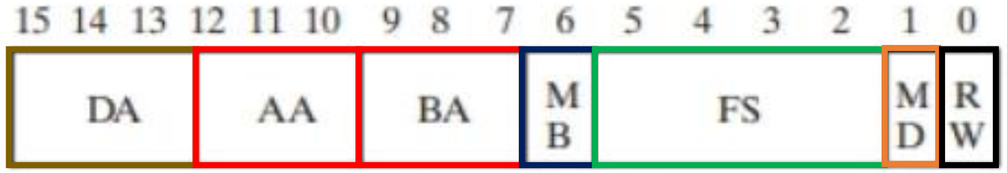



(a) Block diagram

Now let's suppose to have $2^3 = 8$ registers

# Datapaths: The Control Word (4)

- **DA** identifies the address of the destination register
- **AA** identifies the address of the register for the input A
- **BA** identifies the address of the register for the input B
- **MB** selects either the register selected by the BA or a constant from outside
- **FS** identifies the operation to be executed (ALU or shifter)
- **MD** identifies the output
- **RW** identifies whether to write the output to the destination register



(b) Control word

**Control word:**

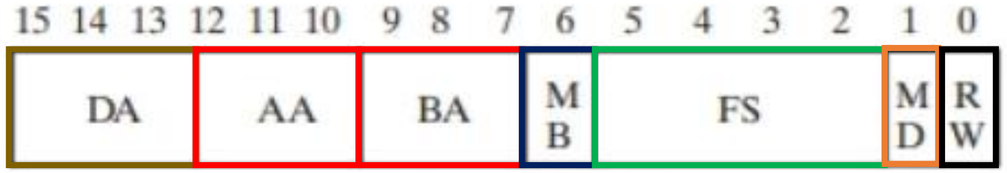
- 16-bit binary control input
- 7 fields
- Identify a microoperation uniquely

Now let's suppose to have

$2^3 = 8$ registers

# Datapaths: The Control Word (4)

- **DA** identifies the address of the destination register
- **AA** identifies the address of the register for the input A
- **BA** identifies the address of the register for the input B
- **MB** selects either the register selected by the BA or a constant from outside
- **FS** identifies the operation to be executed (ALU or shifter)
- **MD** identifies the output
- **RW** identifies whether to write the output to the destination register



(b) Control word

**Control word:**

- 16-bit binary control input
- 7 fields
- Identify a microoperation uniquely

Now let's suppose to have

$2^3 = 8$ registers

# ALU: Arithmetic/Logic Unit

**TABLE 8-2**
**Function Table for ALU**

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

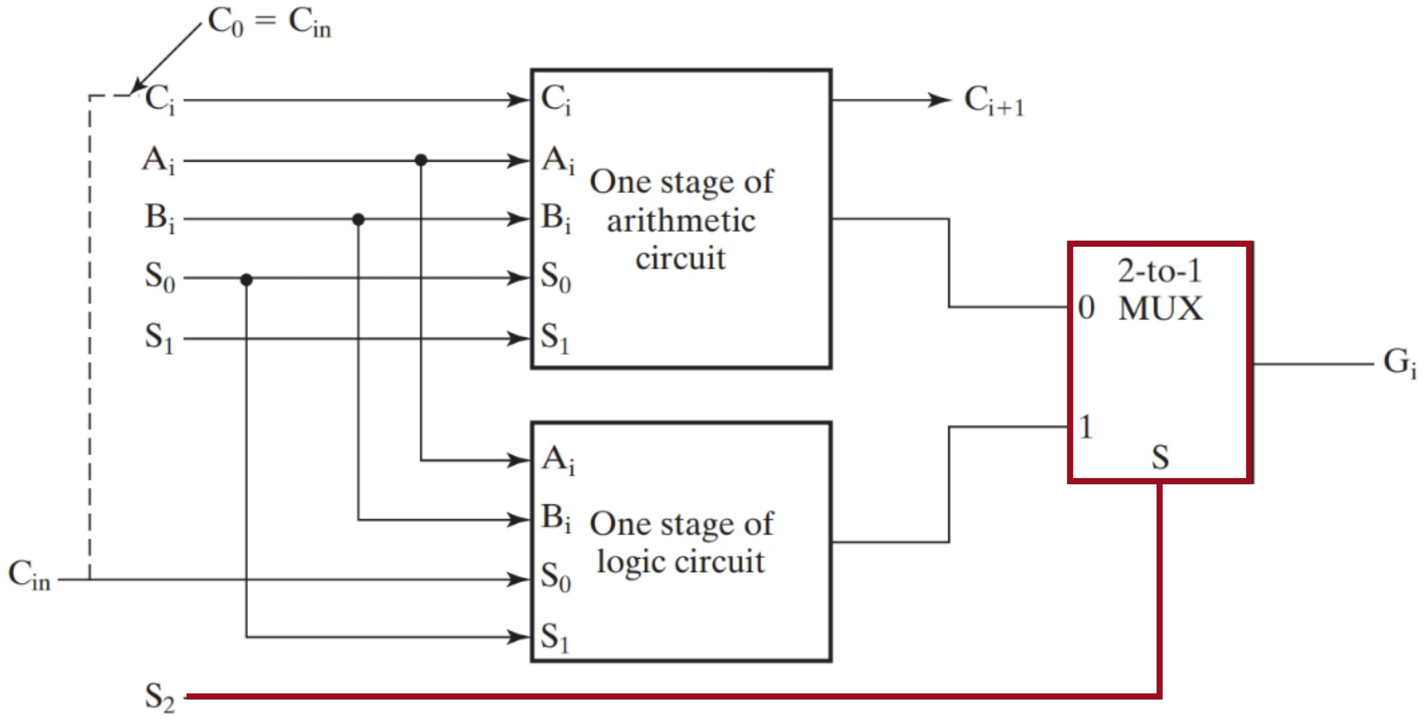
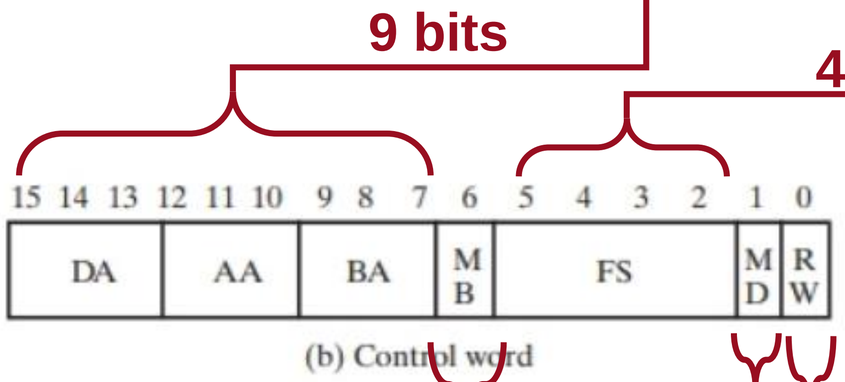



**FIGURE 8-7**
One Stage of ALU

Selection input S1 has no effect during the logic operations and is marked with X to indicate that its value may be either 0 or 1

# Encoding of Control Word for the Datapath

Now let's suppose to have
$2^3 = 8$ registers



(b) Control word

**TABLE 8-5**
**Encoding of Control Word for the Datapath**

| DA, AA, BA | | MB | | FS | | MD | | RW | |
|---|---|---|---|---|---|---|---|---|---|
| Function | Code | Function | Code | Function | Code | Function | Code | Function | Code |
| $R0$ | 000 | Register | 0 | $F = A$ | 0000 | Function | 0 | No Write | 0 |
| $R1$ | 001 | Constant | 1 | $F = A + 1$ | 0001 | Data in | 1 | Write | 1 |
| $R2$ | 010 | | | $F = A + B$ | 0010 | | | | |
| $R3$ | 011 | | | $F = A + B + 1$ | 0011 | | | | |
| $R4$ | 100 | | | $F = A + \overline{B}$ | 0100 | | | | |
| $R5$ | 101 | | | $F = A + \overline{B} + 1$ | 0101 | | | | |
| $R6$ | 110 | | | $F = A - 1$ | 0110 | | | | |
| $R7$ | 111 | | | $F = A$ | 0111 | | | | |
| | | | | $F = A \wedge B$ | 1000 | | | | |
| | | | | $F = A \vee B$ | 1001 | | | | |
| | | | | $F = A \oplus B$ | 1010 | | | | |
| | | | | $F = \overline{A}$ | 1011 | | | | |
| | | | | $F = B$ | 1100 | | | | |
| | | | | $F = \text{sr } B$ | 1101 | | | | |
| | | | | $F = \text{sl } B$ | 1110 | | | | |

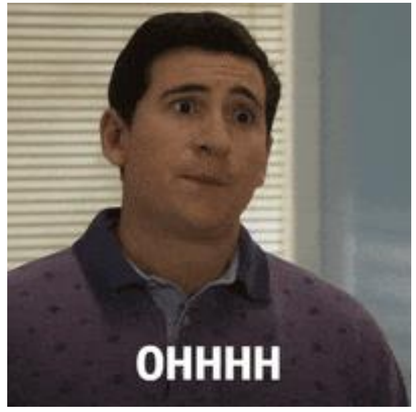
9 bits

4 bits

1 bit

1 bit

1 bit



# Examples of control word (1)

The control word for a given microoperation can be deriving by specifying the value of each of the control fields. For example:

$$R1 \leftarrow R2 + \overline{R3} + 1$$

Destination register

Input A

Input B

Subtraction function

| Field: | DA | AA | BA | MB | FS | MD | RW |
|---|---|---|---|---|---|---|---|
| Symbolic: | $R1$ | $R2$ | $R3$ | Register | $F = A + \overline{B} + 1$ | Function | Write |
| Binary: | 001 | 010 | 011 | 0 | 0101 | 0 | 1 |

| DA | | | AA | | | BA | | | MB | FS | | | | MD | RW |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |

control word

# Examples of control word (2)

**TABLE 8-6**
**Examples of Microoperations for the Datapath, Using Symbolic Notation**

| Micro-operation | DA | AA | BA | MB | FS | MD | RW |
|---|---|---|---|---|---|---|---|
| $R1 \leftarrow R2 - R3$ | $R1$ | $R2$ | $R3$ | Register | $F = A + \overline{B} + I$ | Function | Write |
| $R4 \leftarrow$ sl R6 | $R4$ | _ | $R6$ | Register | $F =$ sl $B$ | Function | Write |
| $R7 \leftarrow R7 + 1$ | $R7$ | $R7$ | _ | _ | $F = A + 1$ | Function | Write |
| $R1 \leftarrow R0 + 2$ | $R1$ | $R0$ | — | Constant | $F = A + B$ | Function | Write |
| Data out $\leftarrow R3$ | _ | _ | $R3$ | Register | _ | _ | No Write |
| $R4 \leftarrow$ Data in | $R4$ | _ | _ | _ | _ | Data in | Write |
| $R5 \leftarrow 0$ | $R5$ | $R0$ | $R0$ | Register | $F = A \oplus B$ | Function | Write |

# Examples of control word (3)

**TABLE 8-7**
**Examples of Microoperations from Table 8-6, Using Binary Control Words**

| Micro-operation | DA | AA | BA | MB | FS | MD | RW |
|---|---|---|---|---|---|---|---|
| $R1 \leftarrow R2 - R3$ | | | | | | | |
| $R4 \leftarrow$ sl R6 | | | | | | | |
| $R7 \leftarrow R7 + 1$ | | | | | | | |
| $R1 \leftarrow R0 + 2$ | | | | | | | |
| Data out $\leftarrow R3$ | | | | | | | |
| $R4 \leftarrow$ Data in | | | | | | | |
| $R5 \leftarrow 0$ | | | | | | | |

# Sequence of microoperations

Assumptions:

- The number of bit in each register is equal to 8
- An unsigned decimal representation is used for all multiplebit signals
- The microoperations are executed in sequence:

**TABLE 8-7**
**Examples of Microoperations from Table 8-6, Using Binary Control Words**

| Micro-operation | DA | AA | BA | MB | FS | MD | RW |
|---|---|---|---|---|---|---|---|
| $R1 \leftarrow R2 - R3$ | 001 | 010 | 011 | 0 | 0101 | 0 | 1 |
| $R4 \leftarrow$ sl R6 | 100 | XXX | 110 | 0 | 1110 | 0 | 1 |
| $R7 \leftarrow R7 + 1$ | 111 | 111 | XXX | X | 0001 | 0 | 1 |
| $R1 \leftarrow R0 + 2$ | 001 | 000 | XXX | 1 | 0010 | 0 | 1 |
| Data out $\leftarrow R3$ | XXX | XXX | 011 | 0 | XXXX | X | 0 |
| $R4 \leftarrow$ Data in | 100 | XXX | XXX | X | XXXX | 1 | 1 |
| $R5 \leftarrow 0$ | 101 | 000 | 000 | 0 | 1010 | 0 | 1 |

**The result of the subtraction in clock cycle 1 appears in register R1 in clock cycle 2**

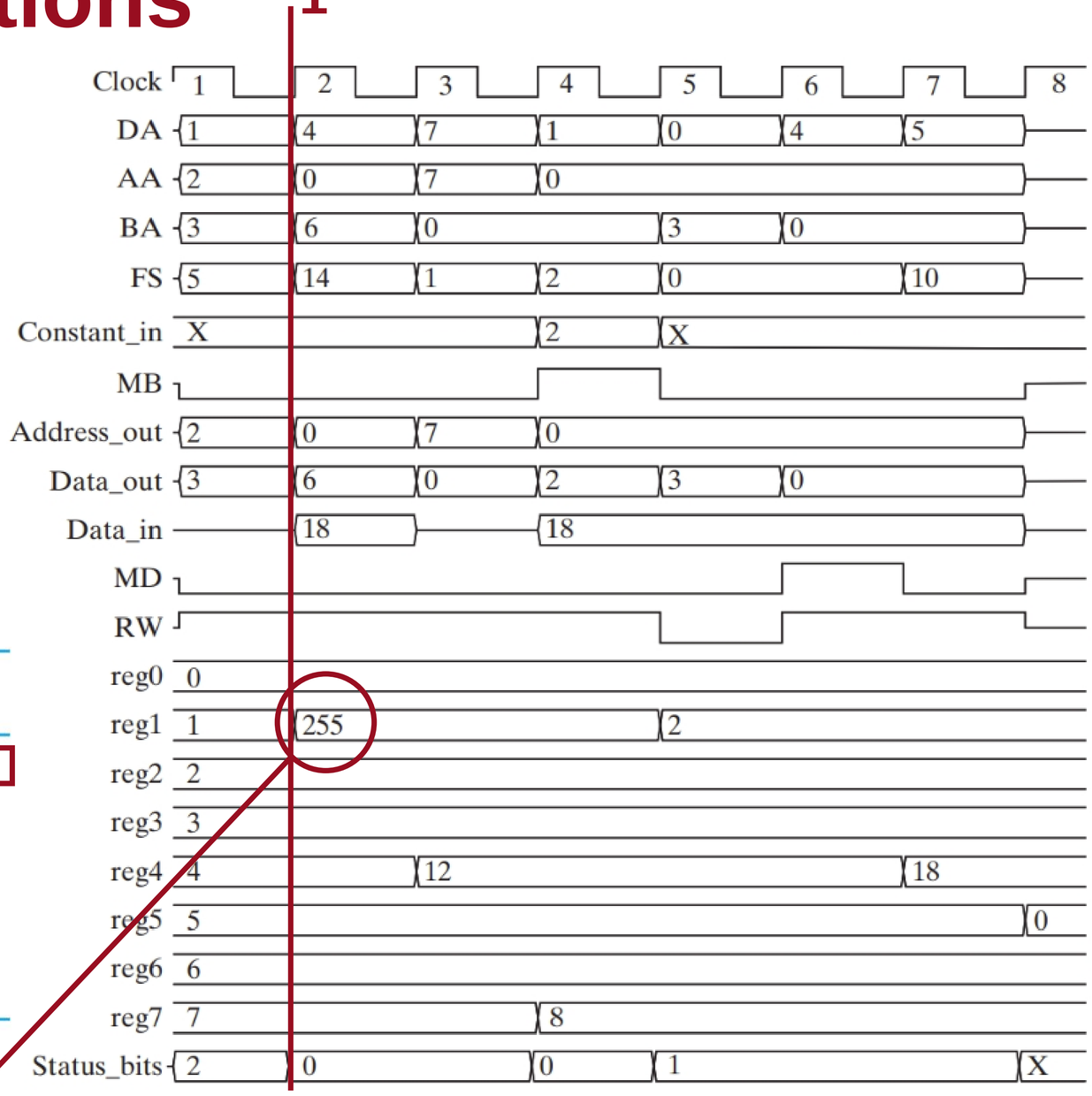



**FIGURE 8-12**
Simulation of the Microoperation Sequence in Table 8-7

# Sequence of microoperations

Assumptions:

- The number of bit in each register is equal to 8
- An unsigned decimal representation is used for all multiplebit signals
- The microoperations are executed in sequence:

**TABLE 8-7**
**Examples of Microoperations from Table 8-6, Using Binary Control Words**

| Micro-operation | DA | AA | BA | MB | FS | MD | RW |
|---|---|---|---|---|---|---|---|
| $R1 \leftarrow R2 - R3$ | 001 | 010 | 011 | 0 | 0101 | 0 | 1 |
| $R4 \leftarrow$ sl R6 | 100 | XXX | 110 | 0 | 1110 | 0 | 1 |
| $R7 \leftarrow R7 + 1$ | 111 | 111 | XXX | X | 0001 | 0 | 1 |
| $R1 \leftarrow R0 + 2$ | 001 | 000 | XXX | 1 | 0010 | 0 | 1 |
| Data out $\leftarrow R3$ | XXX | XXX | 011 | 0 | XXXX | X | 0 |
| $R4 \leftarrow$ Data in | 100 | XXX | XXX | X | XXXX | 1 | 1 |
| $R5 \leftarrow 0$ | 101 | 000 | 000 | 0 | 1010 | 0 | 1 |

**Result available in R4 at the next clock cycle**

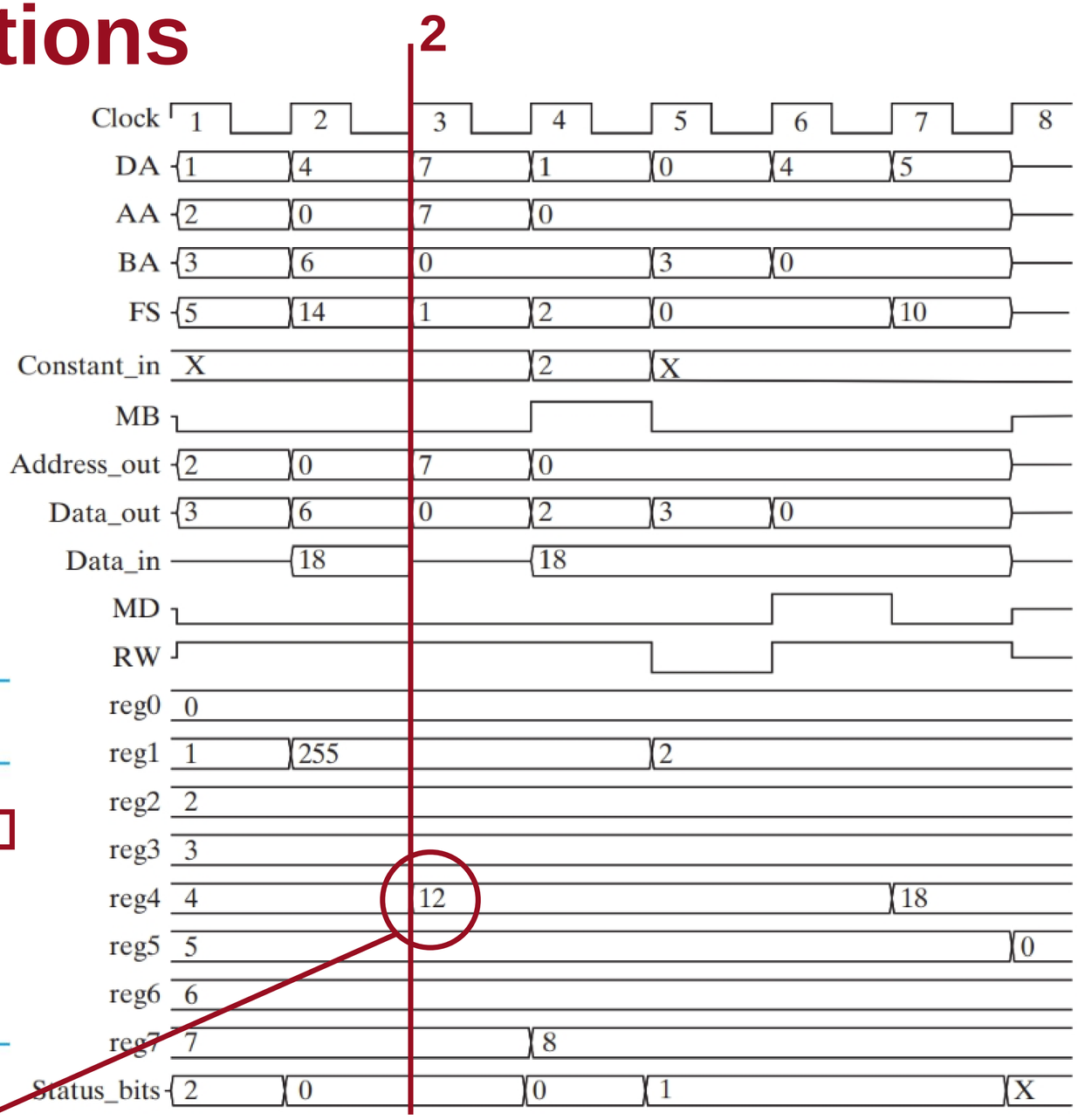



**FIGURE 8-12**
Simulation of the Microoperation Sequence in Table 8-7

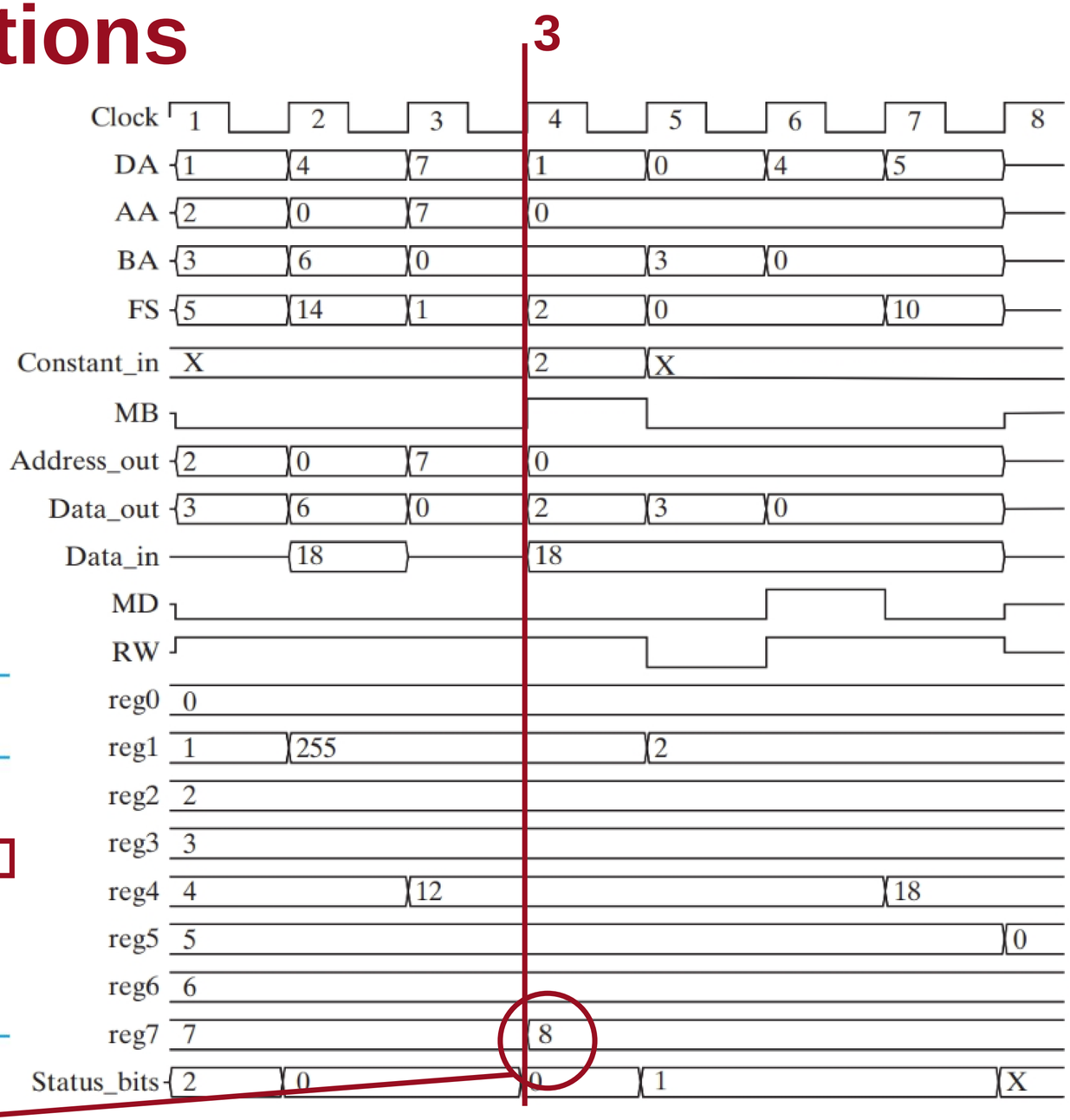
# Sequence of microoperations

Assumptions:

- The number of bit in each register is equal to 8
- An unsigned decimal representation is used for all multiplebit signals
- The microoperations are executed in sequence:

**TABLE 8-7**
**Examples of Microoperations from Table 8-6, Using Binary Control Words**

| Micro-operation | DA | AA | BA | MB | FS | MD | RW |
|---|---|---|---|---|---|---|---|
| $R1 \leftarrow R2 - R3$ | 001 | 010 | 011 | 0 | 0101 | 0 | 1 |
| $R4 \leftarrow$ sl R6 | 100 | XXX | 110 | 0 | 1110 | 0 | 1 |
| $R7 \leftarrow R7 + 1$ | 111 | 111 | XXX | X | 0001 | 0 | 1 |
| $R1 \leftarrow R0 + 2$ | 001 | 000 | XXX | 1 | 0010 | 0 | 1 |
| Data out $\leftarrow R3$ | XXX | XXX | 011 | 0 | XXXX | X | 0 |
| $R4 \leftarrow$ Data in | 100 | XXX | XXX | X | XXXX | 1 | 1 |
| $R5 \leftarrow 0$ | 101 | 000 | 000 | 0 | 1010 | 0 | 1 |

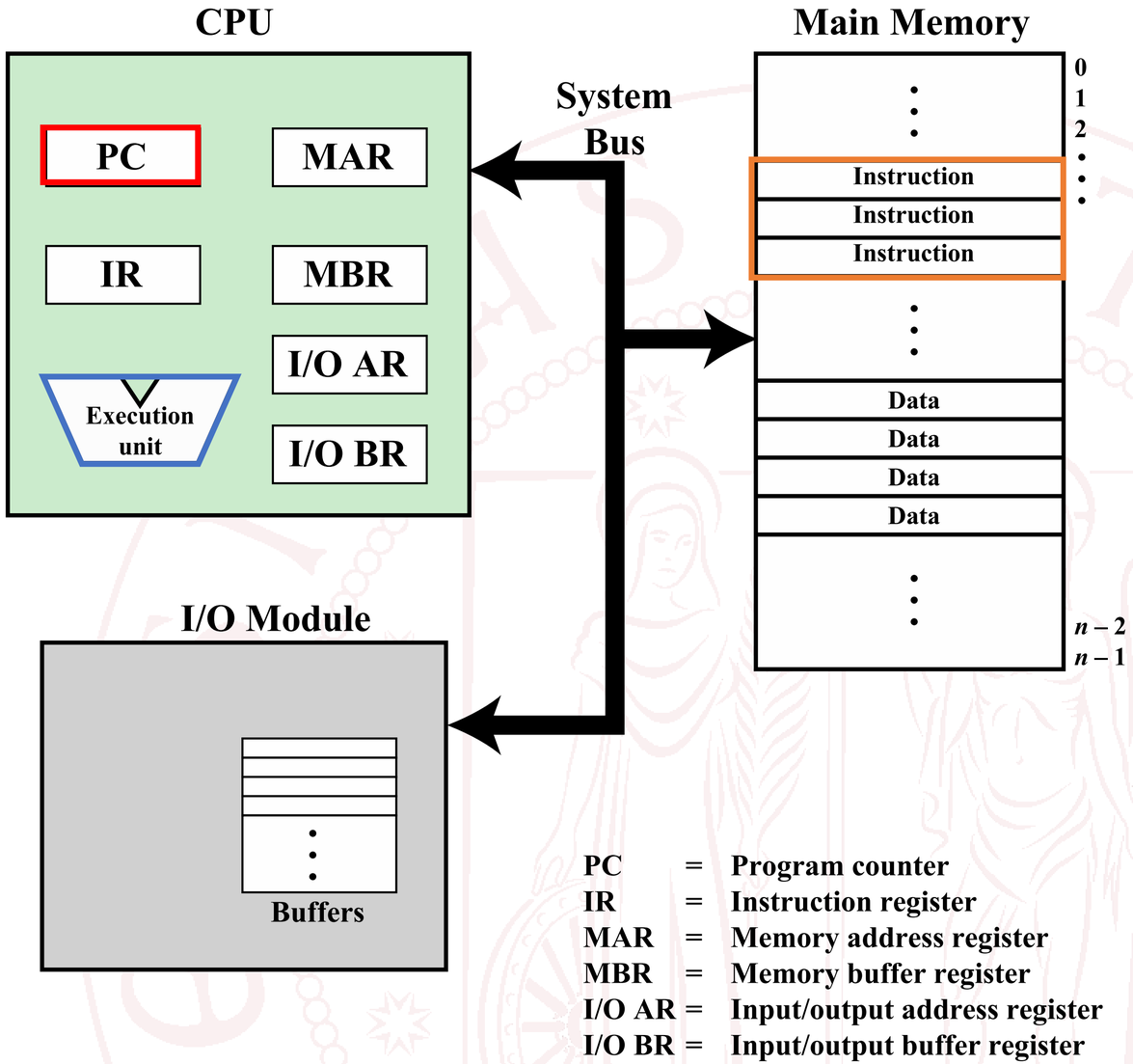



**Result available in R7 at the next clock cycle**

**FIGURE 8-12**
Simulation of the Microoperation Sequence in Table 8-7

# Instructions and e microoperations

- In a programmable system, a portion of the input to the processor consists of a sequence of instructions
- Each instruction specifies:
  - an operation the system has to perform
  - which operands to use
  - where to place the results
  - (which instruction to execute next)
- The instructions are usually stored in memory and they are loaded in sequence into registers.
- To execute the instructions in sequence, the register called Program Counter (PC) provides the address in memory of the instruction to be executed
- A program is a list of instructions



# Instruction Set Architecture (ISA)

- An *instruction* is a collection of bits that instructs the computer to perform a specific operation.
- The collection of instructions for a computer is called **Instruction Set**
- A detailed description of the instruction set provides an abstract model of the computer

  that is called **Instruction Set Architecture (ISA)**
- The simplest ISA has three major components:
  - Storage resources
  - Instruction formats
  - Instruction specifications

# Storage resources

**Resource available to the programmer:**

- Instruction memory: where the instructions are stored
- Data memory: where data are stored
- Register file: registers where the data are saved inside the processor
- Program counter: provides the address in memory of the instruction to be executed

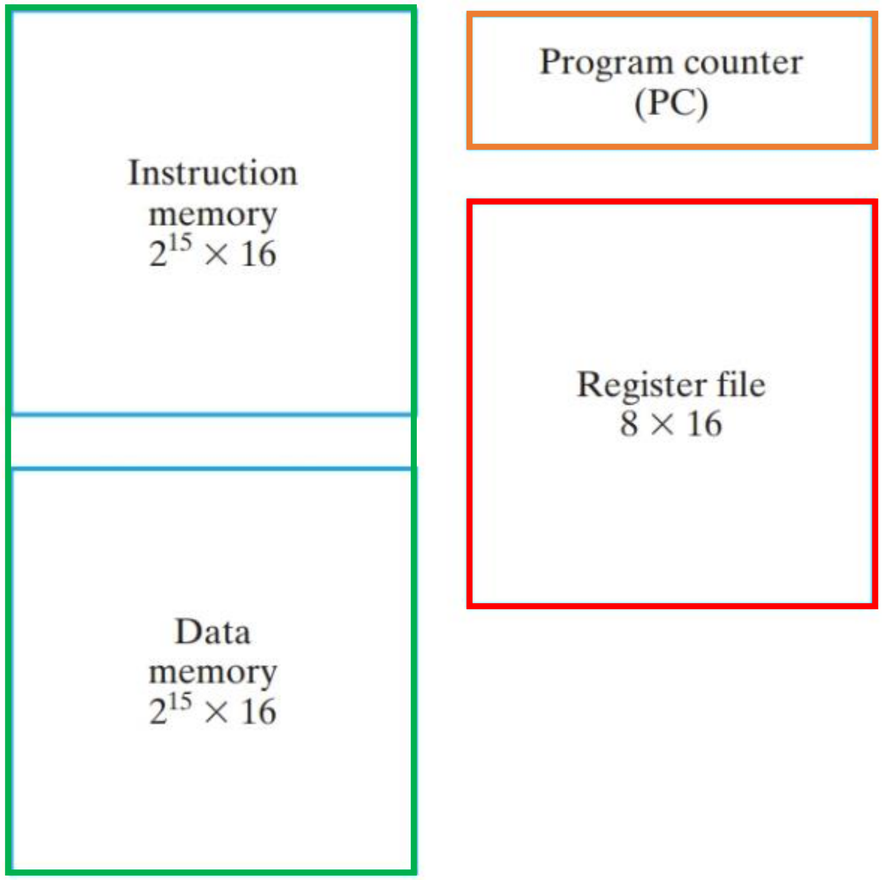
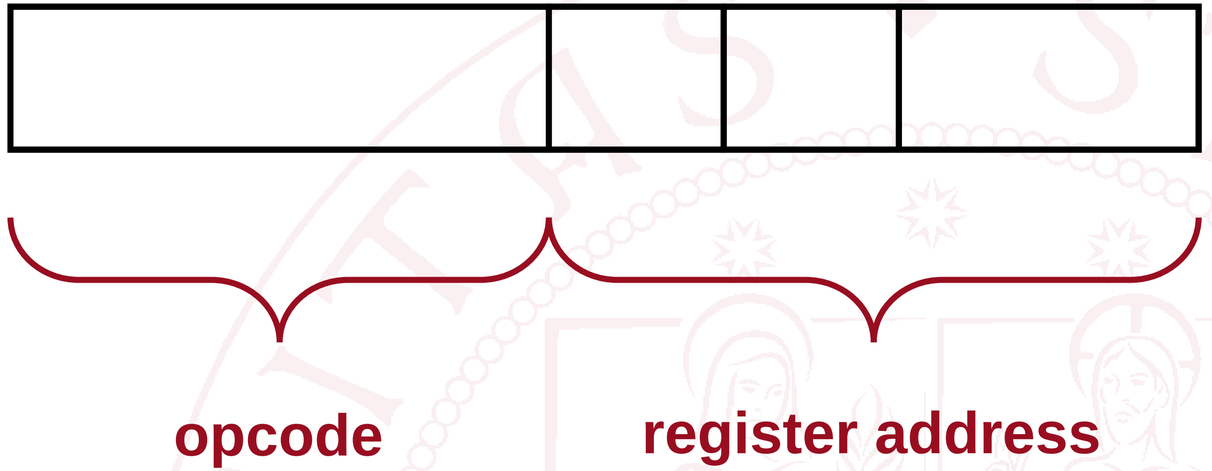



**FIGURE 8-13**
Storage Resource Diagram for a Simple Computer

# Instruction formats

- The format of an instruction is usually depicted by a rectangular box symbolizing the bits of the instruction
- The representation of the bits is as they appear in memory words or in a control register
- The instruction is divided into fields that specify:
  - The operation code (opcode)
  - Register file address



# Instruction formats: operation code (opcode)

- The opcode is a group of bits in the instruction that specifies the operation to be performed (e.g., add, sub, shift..)

- The number of bits required for the opcode of an instruction depends on the total number of distinct operations in the instruction set
($m$ bits for up to $2^m$ distinct operations)

- The designer assigns a bit combination (code) to each operation.


For example:

- Processor with128 distinct operations
- $2^m = 128 \rightarrow m = 7$ → opcode with 7 bit
- opcode: 0000010
- When the opcode is detected by the control unit, a sequence of control words is applied to the datapath

# Instruction formats: register addresses, operands

- The operation must be performed using data stored in computer registers or in a memory
- An instruction, therefore, must specify not only the operation, but also the **operands** and where the **result** has to be placed
- The operands may be specified in two ways:
  - explicitly, if the instruction contains special bits for its identification
  - implicitly, if it is included as a part of the definition of the operation itself (represented by the opcode)


**Explicit operands**

For instance, the instruction performing an addition ($R1 \leftarrow A + B$) may contain:

- 2 registers for the two operands ($A, B$)
- 1 register for the result ($R1$)

**Implicit operands**

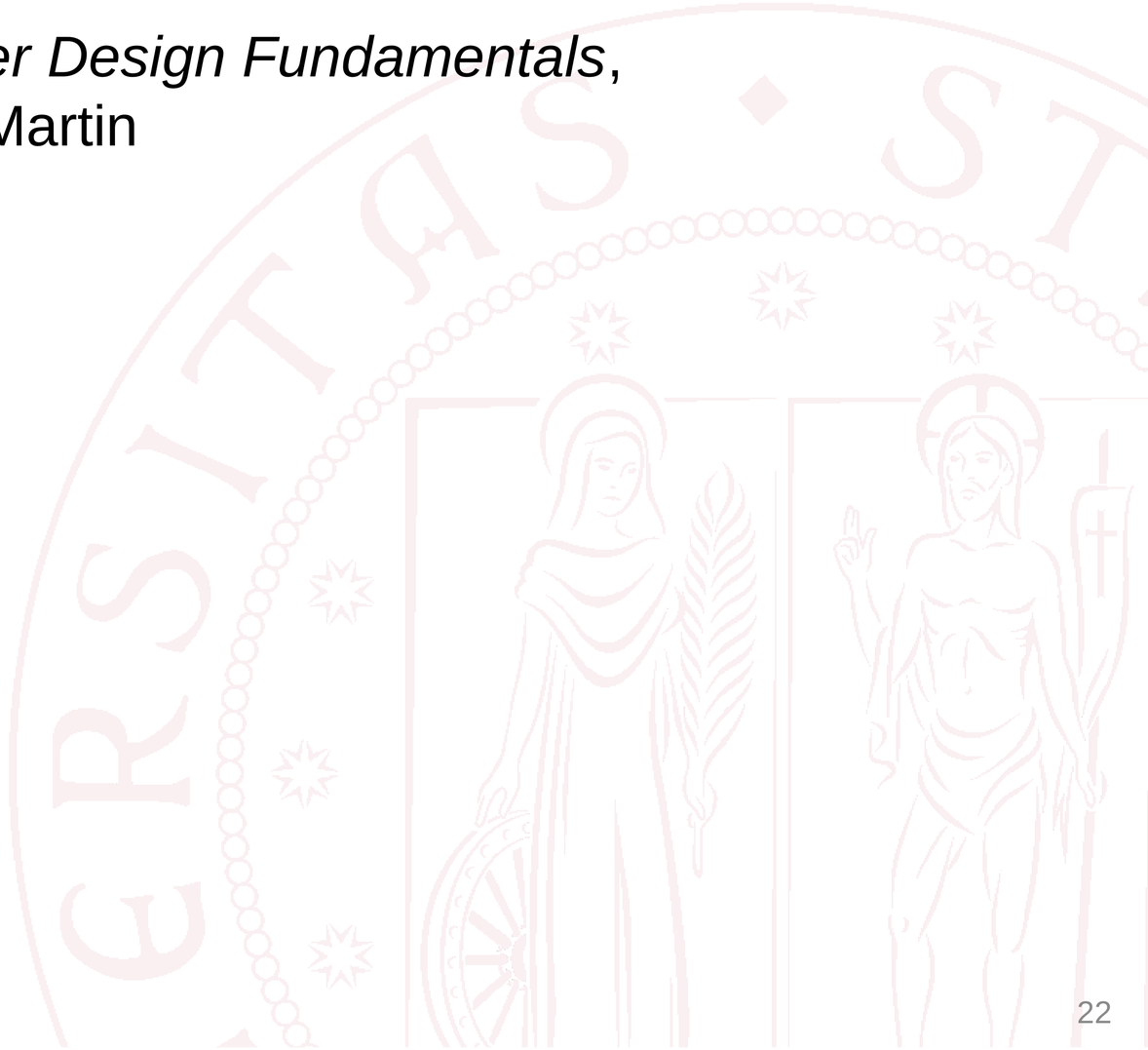
For instance, in an Increment Register operation ($R7 \leftarrow R7 + 1$),

one of the operands is implicitly $+1$

# Disclaimer

Figures from *Logic and Computer Design Fundamentals*,
Fifth Edition, GE Mano | Kime | Martin





# Questions