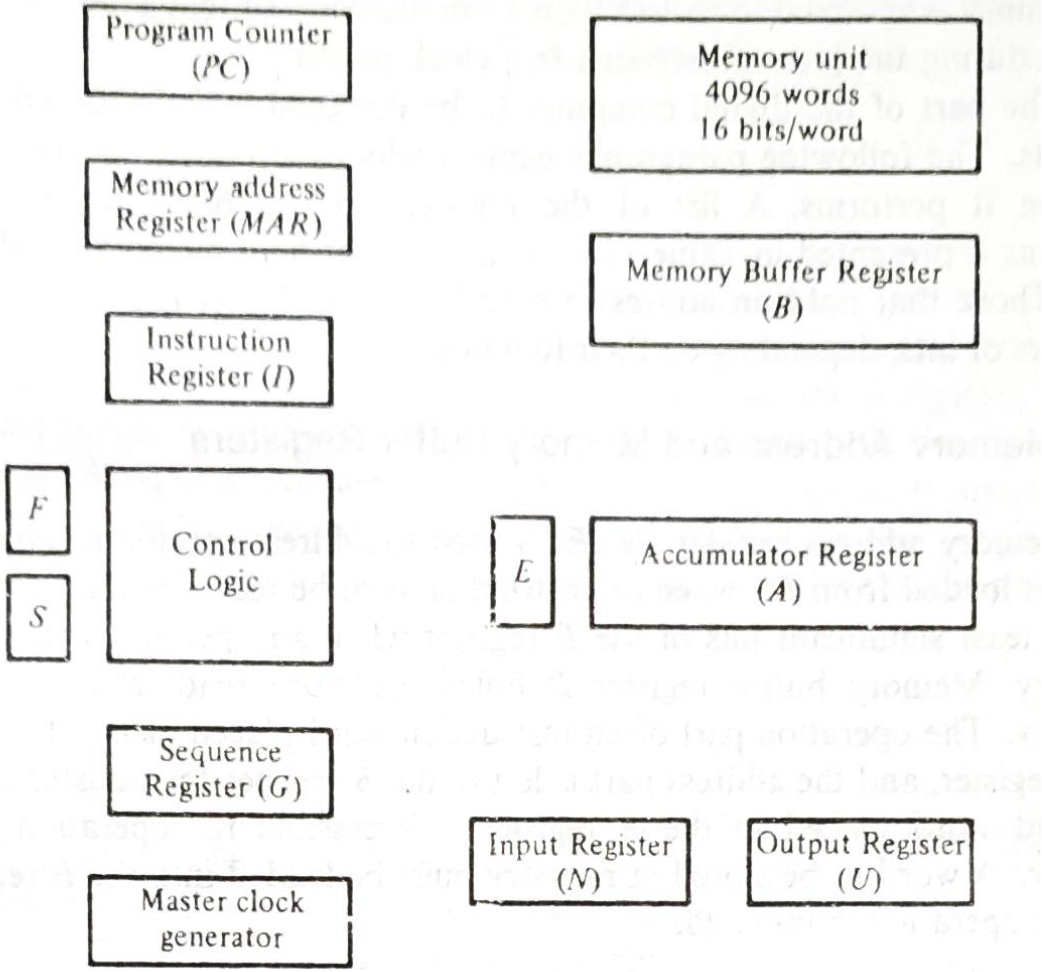
# Digital Systems Design

Computer Design

# Introduction

- The computer consists of a central processor unit, a memory unit and an input-output unit.
- The logic design of the central processor unit will be derived here.
- The design process is divided into six phases:
  - The decomposition of the digital computer into registers which specify the general configuration of the system.
  - The specification of computer instructions.
  - The formulation of a timing and control network.
  - The listing of the register-transfer operations needed to execute all computer instructions.
  - The design of the processor section.
  - The design of the control section.

# System Configuration



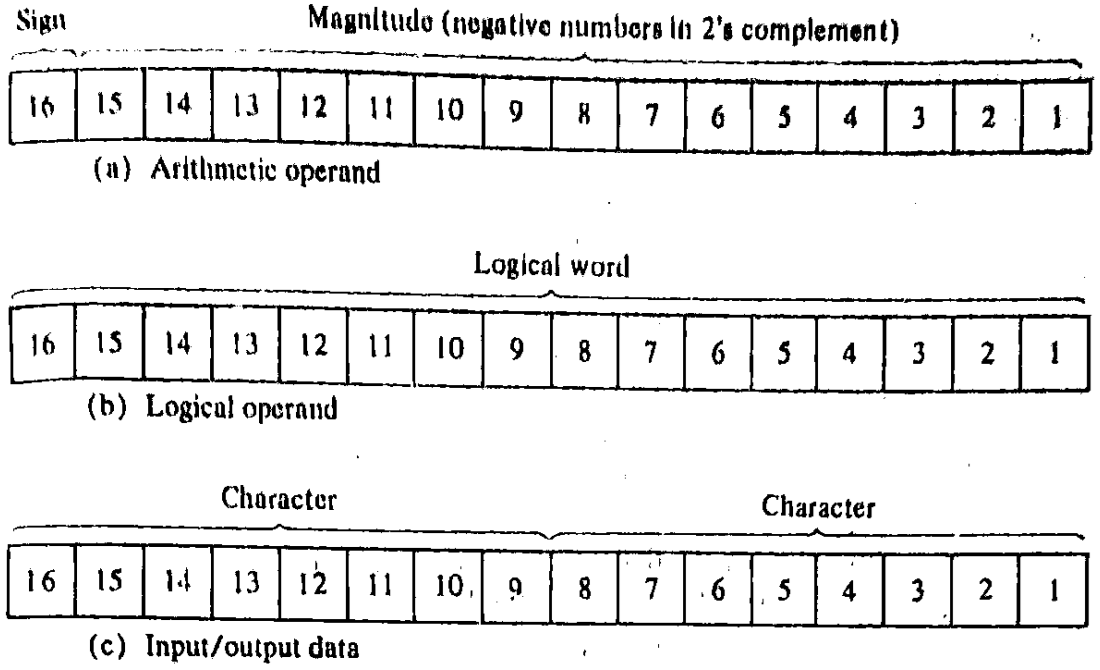

Master-clock generator is a common clock-pulse source which generates a periodic train of pulses.

| Symbolic designation | Name | Number of bits | Function |
|---|---|---|---|
| *A* | Accumulator register | 16 | Processor register |
| *B* | Memory buffer register | 16 | Holds contents of memory word |
| *PC* | Program counter | 12 | Holds address of next instruction |
| *MAR* | Memory address register | 12 | Holds address of memory word |
| *I* | Instruction register | 4 | Holds current operation-code |
| *E* | Extension flip-flop | 1 | Accumulator extension |
| *F* | Fetch flip-flop | 1 | Controls fetch and execute cycles |
| *S* | Start-stop flip-flop | 1 | Starts and stops computer |
| *G* | Sequence register | 2 | Provides timing signals |
| *N* | Input register | 9 | Holds information from input device |
| *U* | Output register | 9 | Holds information for output device |

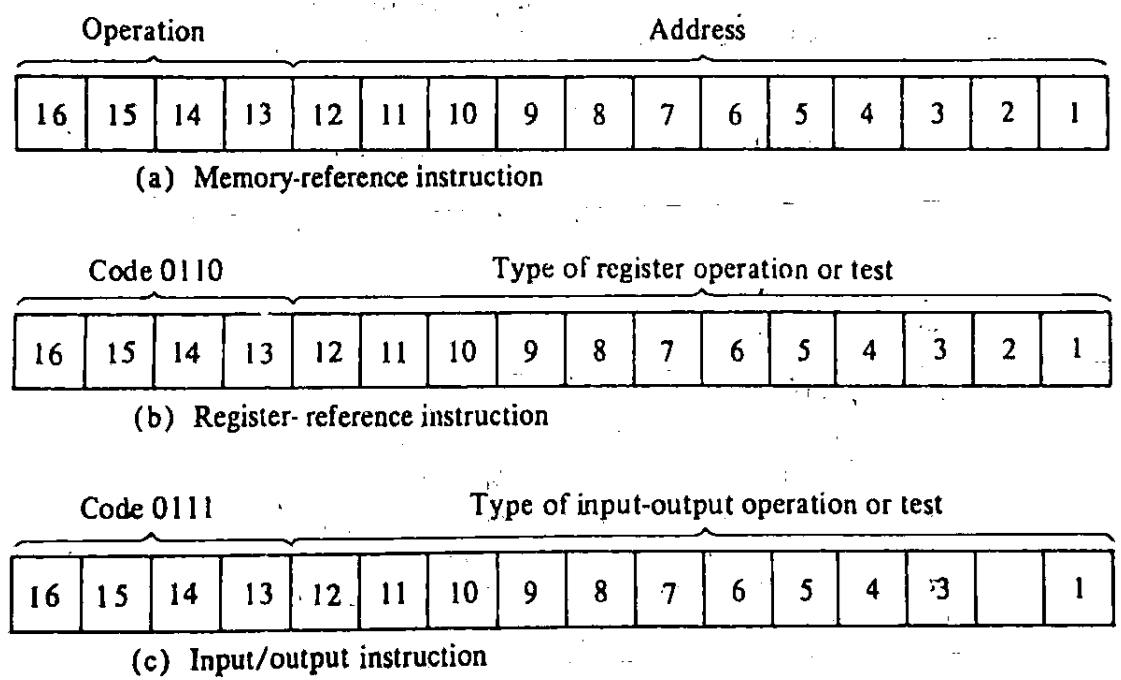
# Computer Instructions

- The instructions must be chosen carefully to supply sufficient capabilities to the system for sovling a wide range of data processing problems.

## Data Formats




## Instruction Formats

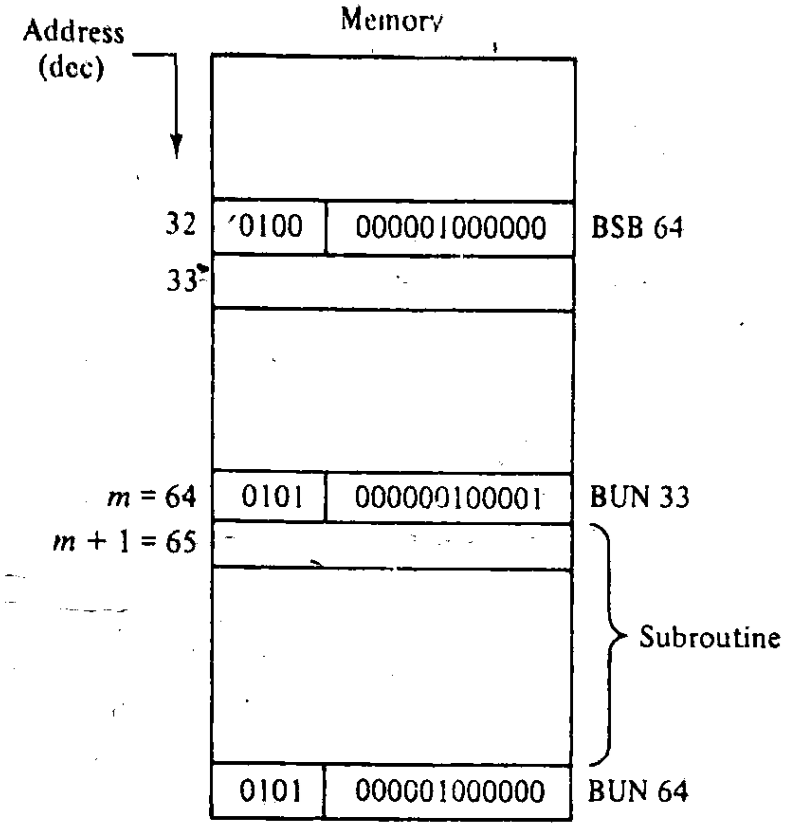


# Computer Instructions
## Memory-reference Instructions

- Six memory-reference instructions for the computer are listed in the table.
- The hexadecimal code listed is an equivalent hexadecimal number of the binary code adopted for the operation-code.

| Symbol | Hexa-decimal code | Description | Function |
|---|---|---|---|
| AND | 0 *m** | AND to *A* | $A \leftarrow A \wedge M^*$ |
| ADD | 1 *m* | Add to *A* | $A \leftarrow A + M$, $E \leftarrow$ Carry |
| STO | 2 *m* | Store in *A* | $M \leftarrow A$ |
| ISZ | 3 *m* | Increment and skip if zero | $M \leftarrow M + 1$, if $(M + 1 = 0)$ then $(PC \leftarrow PC + 1)$ |
| BSB | 4 *m* | Branch to subroutine | $M \leftarrow PC + 5000$, $PC \leftarrow m + 1$ |
| BUN | 5 *m* | Branch unconditionally | $PC \leftarrow m$ |

*m is the address part of the instruction. *M* is the memory word addressed by *m*.



# Computer Instructions
## Register-reference Instructions

- Each register-reference instruction has an operation-code 0110 (hexadecimal 6) and contains a single 1 in one of the remaining 12 bits of the instruction.

| Symbol | Hexa-decimal code | Description | Function |
|---|---|---|---|
| CLA | 6800 | Clear $A$ | $A \leftarrow 0$ |
| CLE | 6400 | Clear $E$ | $E \leftarrow 0$ |
| CMA | 6200 | Complement $A$ | $A \leftarrow \bar{A}$ |
| CME | 6100 | Complement $E$ | $E \leftarrow \bar{E}$ |
| SHR | 6080 | Shift-right $A$ and $E$ | $A \leftarrow \text{shr } A, A_{16} \leftarrow E, E \leftarrow A_1$ |
| SHL | 6040 | Shift-left $A$ and $E$ | $A \leftarrow \text{shl } A, A_1 \leftarrow E, E \leftarrow A_{16}$ |
| INC | 6020 | Increment $A$ | $A \leftarrow A + 1$ |
| SPA | 6010 | Skip on positive $A$ | If $(A_{16} = 0)$ then $(PC \leftarrow PC + 1)$ |
| SNA | 6008 | Skip on negative $A$ | If $(A_{16} = 1)$ then $(PC \leftarrow PC + 1)$ |
| SZA | 6004 | Skip on zero $A$ | If $(A = 0)$ then $(PC \leftarrow PC + 1)$ |
| SZE | 6002 | Skip on zero $E$ | If $(E = 0)$ then $(PC \leftarrow PC + 1)$ |
| HLT | 6001 | Halt computer | $S \leftarrow 0$ |

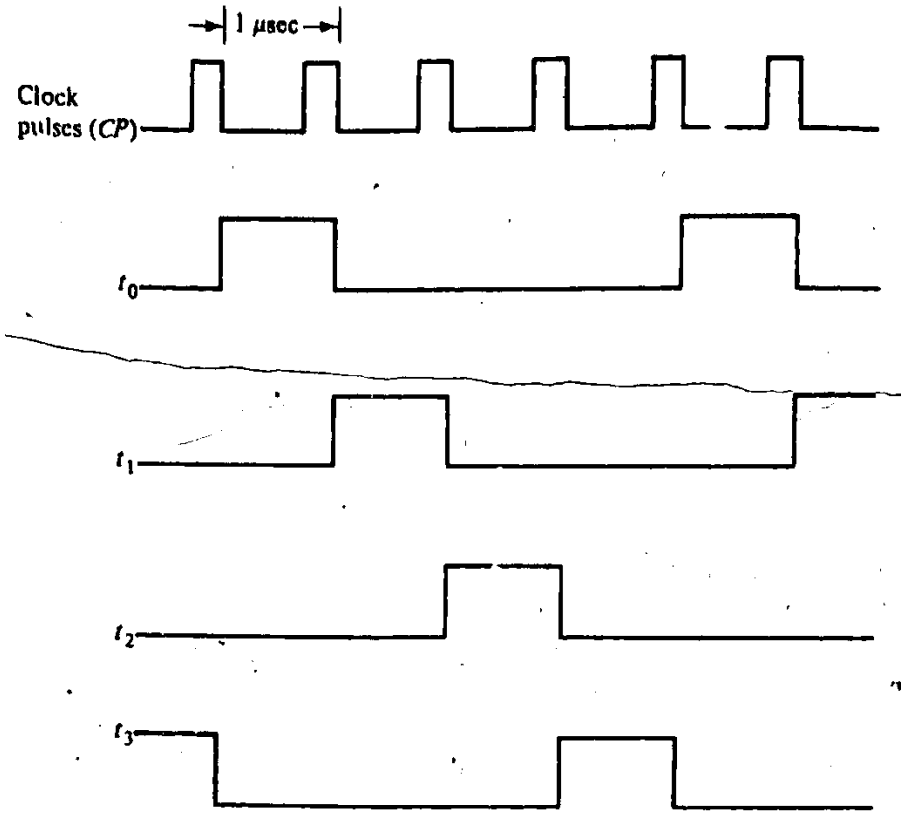
# Computer Instructions
## Input-Output Instructions

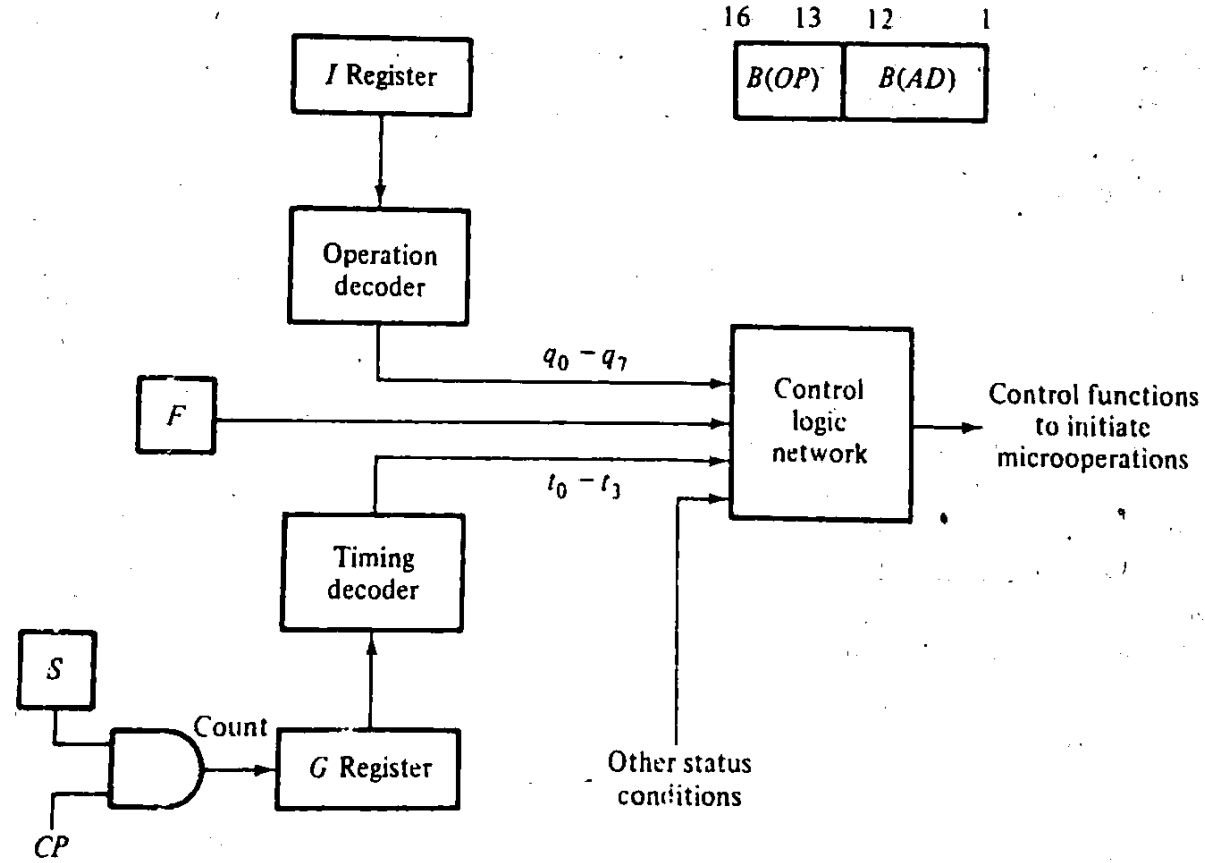
- The computer has four input-output instructions and they are listed in the table.
- These instructions have an operation code 0111 (hexadecimal 7) and each contains a single 1 in one of the remaining 12 bits of the instruction.

| Symbol | Hexadecimal code | Description | Function |
|---|---|---|---|
| SKI | 7800 | Skip on input flag | If ($N_9 = 1$) then ($PC \leftarrow PC + 1$) |
| INP | 7400 | Input to $A$ | $A_{1-8} \leftarrow N_{1-8}$, $N_9 \leftarrow 0$ |
| SKO | 7200 | Skip on output flag | If ($U_9 = 1$) then ($PC \leftarrow PC + 1$) |
| OUT | 7100 | Output from $A$ | $U_{1-8} \leftarrow A_{1-8}$, $U_9 \leftarrow 0$ |

# Timing and Control

- A certain number of timing variables are available in the control unit to sequence the operation in the proper order.



# Execution of Instructions

- Once a “start” switch is activated, the computer sequence follows a basic pattern.
  - An instruction whose address is in *PC* is read from memory.
  - Its operation part is transferred to register *I* and *PC* is incremented by 1 to prepare it for the next instruction.
  - The words read from memory into the *B* register can be either instructions or data.
  - Flip-flop *F* determines:
    - when $F = 0$, it is an instruction and cycle is instruction fetch cycle.
    - when $F = 1$, it is data and cycle is data execution cycle.

# Execution of Instructions
## Fetch Cycle

- An instruction is read from memory during the fetch cycle.
- The operation code in the *I* register is decoded at time $t_3$.
- When an operation code 0, 1, 2, 3, or 4 is encountered, the computer nas to go to an execute cycle to access the memory again. This condition is detected by:

$$F't_0: \quad MAR \leftarrow PC$$
$$F't_1: \quad B \leftarrow M,\ PC \leftarrow PC + 1$$
$$F't_2: \quad I \leftarrow B(OP)$$

$$F'(q_0 + q_1 + q_2 + q_3 + q_4)t_3: \quad F \leftarrow 1$$

| | | |
|---|---|---|
| $F't_0$: | $MAR \leftarrow PC$ | Transfer instruction address |
| $F't_1$: | $B \leftarrow M,\ PC \leftarrow PC + 1$ | Read instruction, increment *PC* |
| $F't_2$: | $I \leftarrow B(OP)$ | Transfer operation code |
| $F'(q_0 + q_1 + q_2 + q_3 + q_4)t_3$: | $F \leftarrow 1$ | Go to execute cycle |
| $q_5t_3$: | $PC \leftarrow B(AD)$ | Branch unconditionally (BUN) |
| $q_6t_3$: | See Table 11-8 | Register-reference instruction |
| $q_7t_3$: | See Table 11-9 | Input-output instruction |

# Execution of Instructions
## Execute Cycle

- At the end of the fetch cycle, the address part of *B* register is transferred to *MAR*.

$$Ft_0: MAR \leftarrow B(AD)$$

$$F(q_0 + q_1 + q_3)t_1: B \leftarrow M$$

- The particular decoded instruction is executed with timing variables $t_2$ and $t_3$. At $t_3$

$$Ft_3: F \leftarrow 0$$

| | | |
|---|---|---|
| $Ft_0$: | $MAR \leftarrow B(AD)$ | Transfer address part |
| $F(q_0 + q_1 + q_3)t_1$: | $B \leftarrow M$ | Read operand |
| $F(t_2 + t_3)$: | See Table 11-7 | Execute memory-reference instruction |
| $Ft_3$: | $F \leftarrow 0$ | Return to fetch cycle |

# Execution of Instructions

## Execute Cycle

| | | | |
|---|---|---|---|
| AND | $Fq_0t_3$: | $A \leftarrow A \wedge B$ | AND microoperation |
| ADD | $Fq_1t_3$: | $A \leftarrow A + B$, $E \leftarrow$ carry | Add microoperation |
| STO | $Fq_2t_2$: | $B \leftarrow A$ | Transfer $A$ to $B$ |
| | $Fq_2t_3$: | $M \leftarrow B$ | Store in memory |
| ISZ | $Fq_3t_2$: | $B \leftarrow B + 1$ | Increment memory word |
| | $Fq_3t_3$: | $M \leftarrow B$ | Store back in memory |
| | $Fq_3B_zt_3$: | $PC \leftarrow PC + 1$ | Skip if $B_z = 1$ ($B = 0$) |
| BSB | $Fq_4t_2$: | $B(AD) \leftarrow PC$, $B(OP) \leftarrow 0101$, $PC \leftarrow MAR$ | Transfer return address, transfer address to $PC$ |
| | $Fq_4t_3$: | $M \leftarrow B$, $PC \leftarrow PC + 1$ | Store return address, increment address in $PC$ |