Control Unit Operation

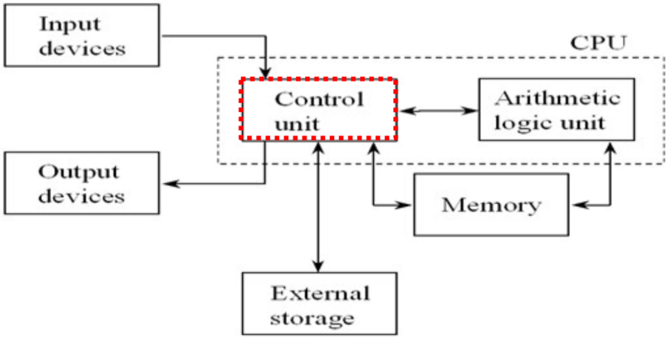
Chapter 20

Joannah Nanjekye

August 08, 2024

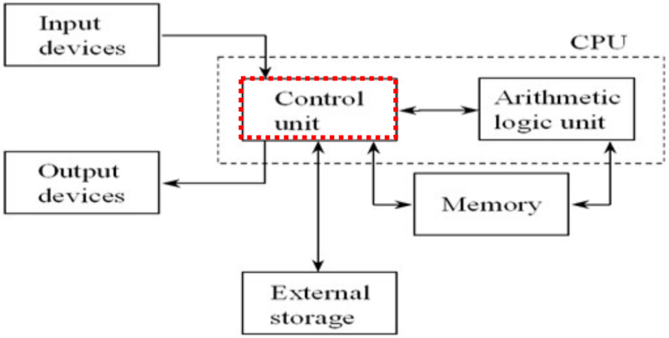
Control Unit

- Part of a CPU or other device that directs its operations
- Tells the rest of the computer system how to carry out a program's instructions



Control Unit Operation

- Circuitry that controls the flow of information through the processor
- Directs the movement of signals between memory and the ALU
- Also directs control signals between the CPU and I/O devices



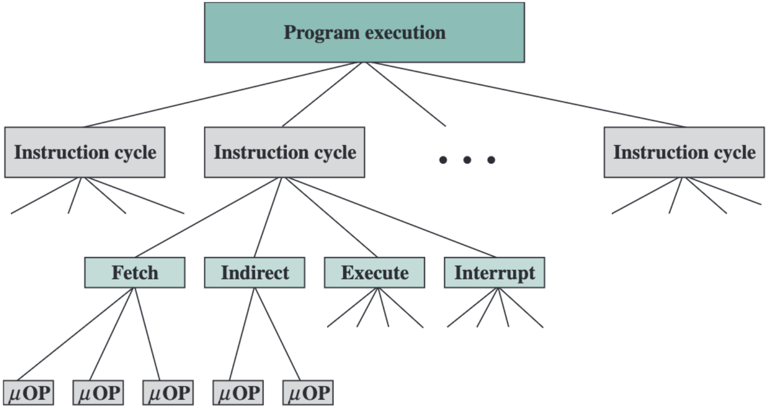
Functional Requirements of a Processor

- Operations (opcodes)
- Addressing modes
- Registers
- I/O module interface
- Memory module interface
- Interrupts

Micro-Operations

- Are the functional or atomic operations of a processor
- A single micro-operation generally involves transfer between:
  - Registers
  - Registers and external bus
- A micro-operation can also be a simple ALU operation

Micro-Operations



The concept of micro-operations serves as a guide to the design of the control unit

Review of Registers

- **Memory address register (MAR):** Is connected to the address lines of the system bus. It specifies the address in memory for a read or write operation
- **Memory buffer register (MBR):** Is connected to the data lines of the system bus. It contains the value to be stored in memory or the last value read from memory
- **Program counter (PC):** Holds the address of the next instruction to be fetched.
- **Instruction register (IR):** Holds the last instruction fetched

Micro-operations

$$
\begin{array}{lllll}
t_1: & \mathtt{MAR} & \leftarrow & \mathtt{(PC)} & \\
t_2: & \mathtt{MBR} & \leftarrow & \mathtt{Memory} & \\
 & \mathtt{PC} & \leftarrow & \mathtt{(PC)} & + \ I \\
t_3: & \mathtt{IR} & \leftarrow & \mathtt{(MBR)} &
\end{array}
$$

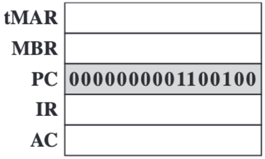
- Each clock pulse defines a time unit
- Each micro-operation can be performed within the time of a single time unit
- The notation (t_1, t_2, t_3) represents successive time units

Rules for Grouping Micro-operations

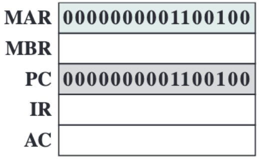
- The proper sequence of events must be followed
  - Thus (MAR $\leftarrow$ (PC)) must precede $(MBR \leftarrow Memory)$
  - Because the memory read operation makes use of the address in the MAR
- Conflicts must be avoided
  - No attempt to read to and write from the same register in one time unit
  - Results get unpredictable
  - E.g,. $(MBR \leftarrow Memory) and (IR \leftarrow MBR)$

Fetch Cycle

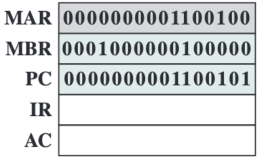
$$\begin{array}{lll} t_1: & \mathtt{MAR} & \leftarrow \; \mathtt{(PC)} \\ t_2: & \mathtt{MBR} & \leftarrow \; \mathtt{Memory} \\ & \mathtt{PC} & \leftarrow \; \mathtt{(PC)} + I \\ t_3: & \mathtt{IR} & \leftarrow \; \mathtt{(MBR)} \end{array}$$



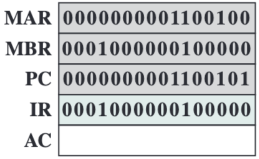
(a) Beginning (before t_1)



(b) After first step



(c) After second step



(d) After third step

The Indirect Cycle

- The address field of the instruction is transferred to the MAR
- The address field of the IR is updated from the MBR
- IR now contains direct addressing

t_1: MAR $\leftarrow$ (IR(Address))
t_2: MBR $\leftarrow$ Memory
t_3: IR(Address) $\leftarrow$ (MBR(Address))

The Interrupt Cycle

- Contents of the PC are transferred to the MBR (facilitate return)
- MAR is loaded with the address at which the contents of the PC
- The PC is loaded with the address of the start of the interrupt-processing routine
- Then store the MBR

$$\begin{array}{lll}
t_1: & \mathtt{MBR} & \leftarrow \mathtt{(PC)} \\
t_2: & \mathtt{MAR} & \leftarrow \mathtt{Save\_Address} \\
 & \mathtt{PC} & \leftarrow \mathtt{Routine\_Address} \\
t_3: & \mathtt{Memory} & \leftarrow \mathtt{(MBR)}
\end{array}$$

The Execute Cycle

- Micro-operations for the execute cycle depend on the opecode
- **Instruction Decoding:** The control unit examines the opcode and generates a sequence of micro-operations based on the value of the opcode

$$\begin{aligned} t_1 &: \text{MAR} \leftarrow (\text{IR(address)}) \\ t_2 &: \text{MBR} \leftarrow \text{Memory} \\ t_3 &: \text{R1} \leftarrow (\text{R1}) + (\text{MBR}) \end{aligned}$$

Example

ADD R1, X

$$
\begin{aligned}
\mathtt{t_1:}\ & \mathtt{MAR} \leftarrow \mathtt{(IR(address))} \\
\mathtt{t_2:}\ & \mathtt{MBR} \leftarrow \mathtt{Memory} \\
\mathtt{t_3:}\ & \mathtt{R1} \leftarrow \mathtt{(R1)} + \mathtt{(MBR)}
\end{aligned}
$$

Example

ISZ X

```
$t_1$: MAR ← (IR(address))
$t_2$: MBR ← Memory
$t_3$: MBR ← (MBR) + 1
$t_4$: Memory ← (MBR)
    If ((MBR) = 0) then (PC ← (PC) + I)
```

Example

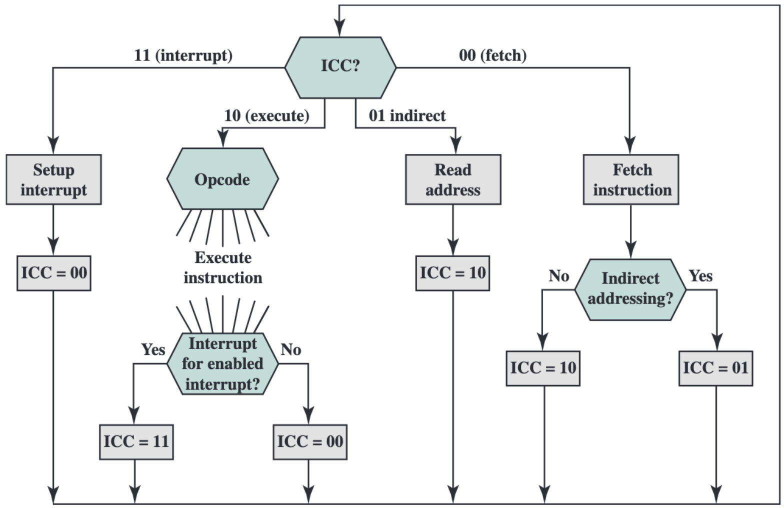
BSA X

t_1: MAR $\leftarrow$ (IR(address))
MBR $\leftarrow$ (PC)
t_2: PC $\leftarrow$ (IR(address))
Memory $\leftarrow$ (MBR)
t_3: PC $\leftarrow$ (PC) + I

Instruction Cycle

- Each phase can be decomposed into a sequence of elementary micro-operations
- E.g fetch, indirect, and interrupt cycles
- Execute cycle:
  - One sequence of micro-operations for each opcode
- Need to tie sequences of micro-operations together
- Assume a new 2-bit register, Instruction cycle code (ICC)
- It designates which part of cycle the processor is in:
  - 00: Fetch
  - 01: Indirect
  - 10: Execute
  - 11: Interrupt

Flow Chart for the Instruction Cycle



Control of the Processor

Key functional requirements:

- Define the basic elements of a processor
- Describe micro-operations that the processor performs
- Determine functions that the control unit must perform

Basic Elements of a Processor

- ALU
- Registers
- Internal data paths
- External data paths
- Control Unit

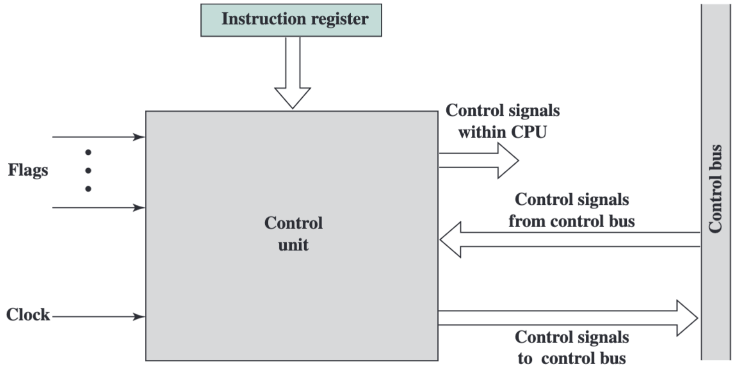
Categorization of Micro-operations

- All of the micro-operations fall into one of the following categories:
  - Transfer data between registers
  - Transfer data from register to external Transfer data from external to register
  - Perform arithmetic or logical operations

Functions of a Control Unit

- **Sequencing**
  - Causing the CPU to step through a series of micro-operations
- **Execution**
  - Causing the performance of each micro-operation
- Both functions are achieved using control signals

General Model of the Control Unit


Instruction register
Control signals within CPU
Flags
Control signals from control bus
Control bus
Control unit
Clock
Control signals to control bus

Control Signals - Input

- Clock
  - One micro-instruction per clock cycle
- Instruction Register
  - Ope-code for current instruction determines which micro-instructions re performed during the execution cycle
- Flags
  - State of CPU
  - Results of previous operations
- From control bus
  - Interrupts
  - Acknowledgements

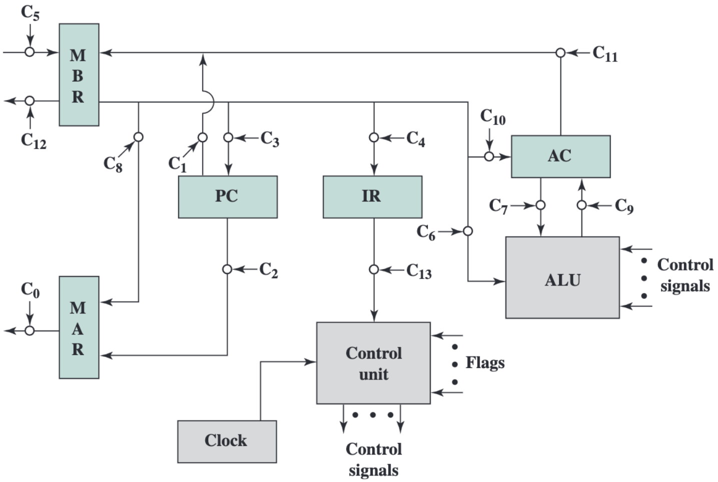
Control Signals - Output

- Within the CPU
  - Cause data movement
  - Activate specific ALU functions
- Via control bus
  - To memory
  - To I/o devices

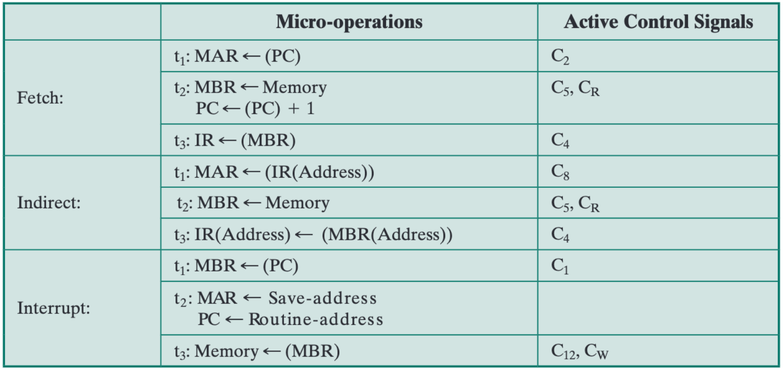
Example: Control Signal Sequence - Fetch

- $MAR \leftarrow (PC)$
  - Control unit activates signal to open gates between the PC and MAR
- $MBR \leftarrow (Memory)$
  - Open gates between MAR and address bus
  - Open gates between data bus and MBR

A Control Signals Example

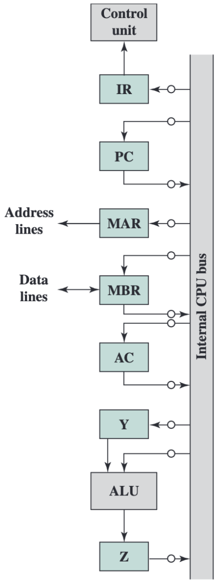


Internal Processor Organization



	Micro-operations	Active Control Signals
Fetch:	t_1: MAR ← (PC)	C_2
	t_2: MBR ← Memory PC ← (PC) + 1	C_5, C_R
	t_3: IR ← (MBR)	C_4
Indirect:	t_1: MAR ← (IR(Address))	C_8
	t_2: MBR ← Memory	C_5, C_R
	t_3: IR(Address) ← (MBR(Address))	C_4
Interrupt:	t_1: MBR ← (PC)	C_1
	t_2: MAR ← Save-address PC ← Routine-address	
	t_3: Memory ← (MBR)	C_{12}, C_W

CPU with Internal Bus

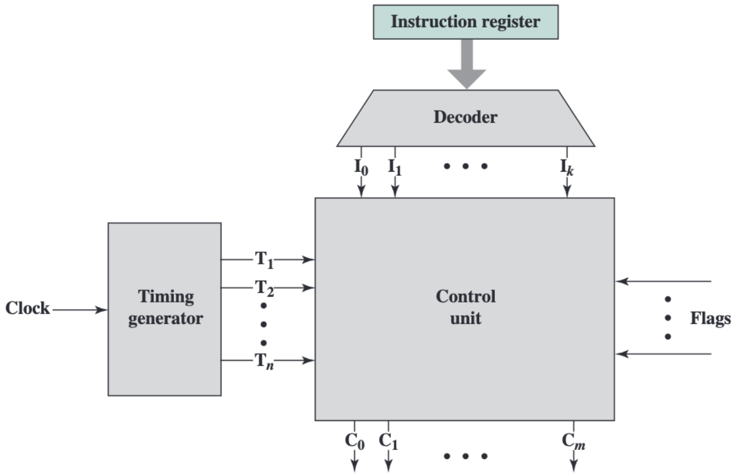


Control Unit Implementation

Two main categories of implementation:

- Microprogrammed implementation
- Hardwired implementation
    - Control unit is a state machine circuit
    - Input logic signals transformed into output logic signals
    - Which are the control signals

Control Unit with Decoded Inputs



Hardwired implementation

- **Control Unit Inputs**
  - Flags and control bus
    - Each bit means something
  - Instruction register
    - ope-code cause different control signals for each different instruction
    - Decoder takes encoded input and produces single output
    - n binary inputs and 2^n outputs
- **Control unit logic**
  - Logic of the control unit that produces output control signals as a function of its input signals

Control Unit Logic

Let us consider a single control signal, C_5 . This signal causes data to be read from the external data bus into the MBR

- Let us define two new control signals, P and Q and the boolean expression C_5 defines

$PQ = 00$	Fetch Cycle
$PQ = 01$	Indirect Cycle
$PQ = 10$	Execute Cycle
$PQ = 11$	Interrupt Cycle

$$C_5 = \overline{P} \bullet \overline{Q} \bullet T_2 + \overline{P} \bullet Q \bullet T_2$$

Sources Acknowledgement

- Course Textbook
- https://slideplayer.com/slide/9710526/
- https://slideplayer.com/slide/5855121/
- https://slideplayer.com/slide/6607257/