Computer Organization and Architecture

16 Control Unit Operation

Tongwei Ren

Dec. 10, 2019

南京大學
NANJING UNIVERSITY

Review

- CPU structure
- Register
  - General purpose, control and status
- Indirect cycle
- Data flow
- Pipeline
  - Two stages, six stages, performance, hazard (structure hazard, data hazard, control hazard)

Function of A Processor

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

Defined by the instruction set

Defined by specifying the system bus

Defined partially by system bus and partially by the type of supporting operating system

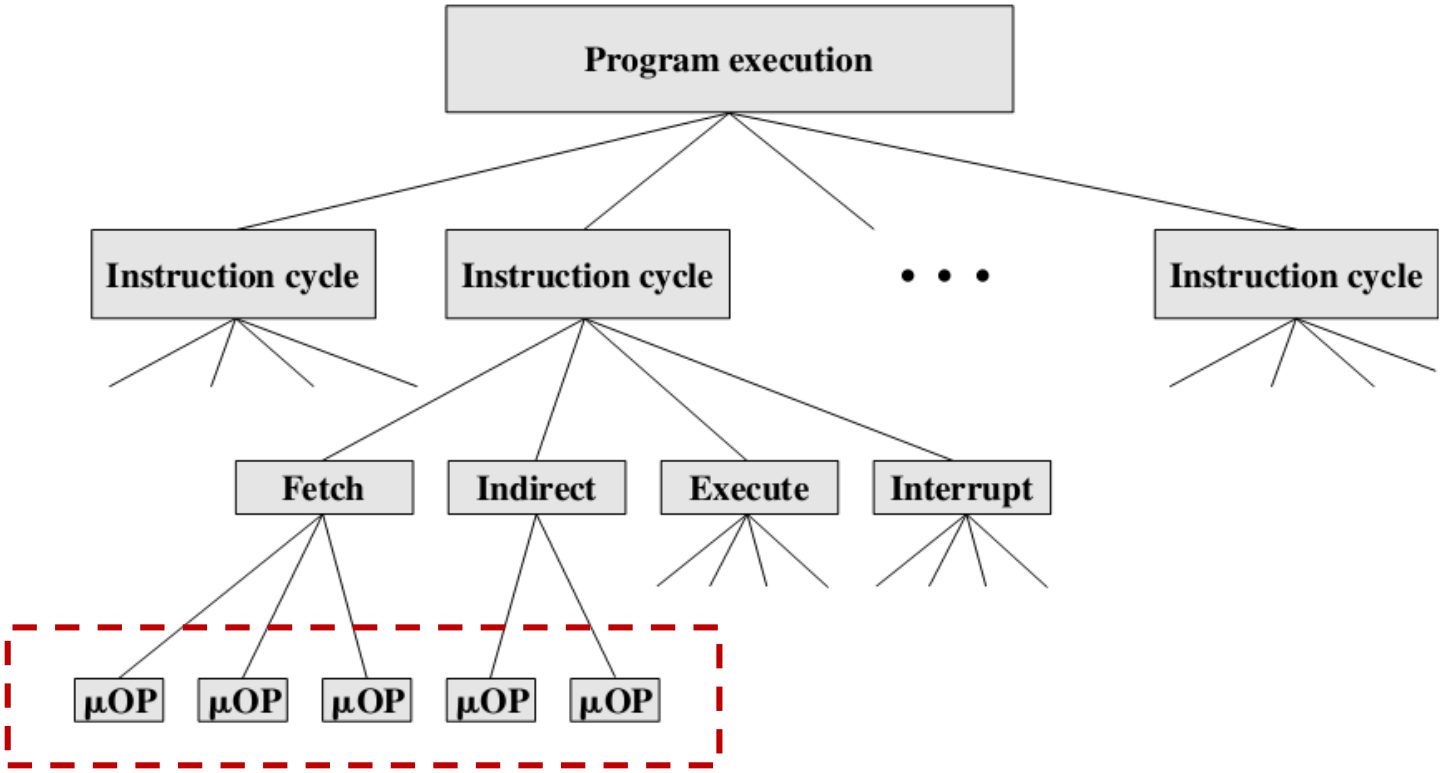
How these functions are performed?

How the various elements of the processor are controlled to provide these functions?

Micro-Operations

- The operation of a computer, in executing a program, consists of a sequence of instruction cycles, with one machine instruction per cycle
- Each instruction cycle is made up of a number of smaller units
  - E.g.: fetch, indirect, execute, and interrupt
- Each of the smaller cycles involves a series of steps, each of which involves the processor registers
  - These steps are referred to as **micro-operations**

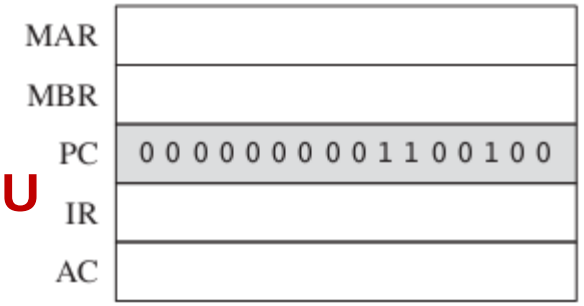
Micro-Operations (cont.)



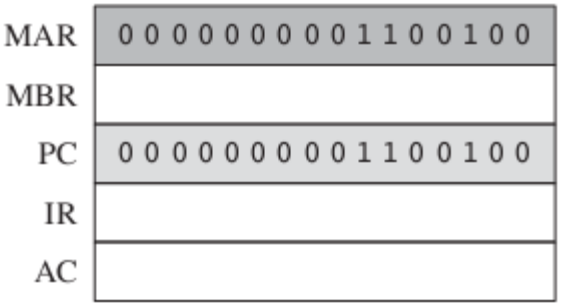
Fetch Cycle

- It occurs at the beginning of each instruction cycle and causes an instruction to be fetched from memory

t_1: MAR ← (PC)
t_2: MBR ← Memory
PC ← (PC) + *I* **ALU**
t_3: IR ← (MBR)

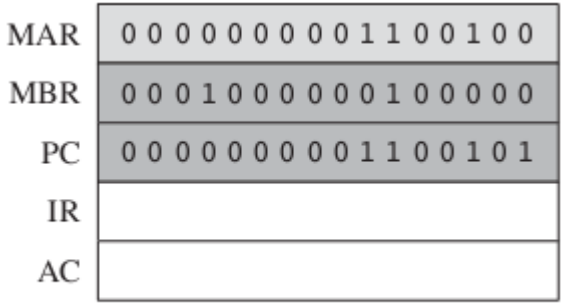


(a) Beginning (before t_1)

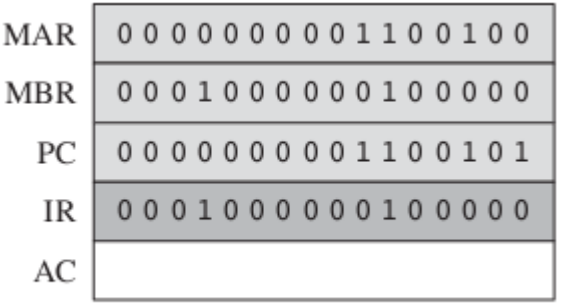


(b) After first step

t_1: MAR ← (PC)
t_2: MBR ← Memory
t_3: PC ← (PC) + *I*
IR ← (MBR)



(c) After second step



(d) After third step

Two Simple Rules of Grouping Micro-operations

- The proper sequence of events must be followed
  - E.g.: (MAR ← (PC)) must precede (MBR ← Memory) because the memory read operation makes use of the address in the MAR
- Conflicts must be avoided
  - E.g.: (MBR ← Memory) and (IR ← MBR) should not occur during the same time unit

Indirect Cycle

- If the instruction specifies an indirect address, then an indirect cycle must precede the execute cycle

t_1: MAR ← (IR(Address))
t_2: MBR ← Memory
t_3: IR(Address) ← (MBR(Address))

- IR is now in the same state as if indirect addressing had not been used, and it is ready for the execute cycle

Interrupt Cycle

- After execute cycle, a test is made to determine whether any enabled interrupts have occurred
- If so, the interrupt cycle occurs

```
t1: MBR ← (PC)
t2: MAR ← Save_Address
    PC ← Routine_Address
t3: Memory ← (MBR)
```

Execute Cycle

- With the variety opcodes, there are a number of different sequences of micro-operations that can occur
- Add instruction

```
ADD R1, X
```

t_1: MAR ← (IR(address))
t_2: MBR ← Memory
t_3: R1 ← (R1) + (MBR)

- Increment and skip if zero

```
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)

Execute Cycle (cont.)

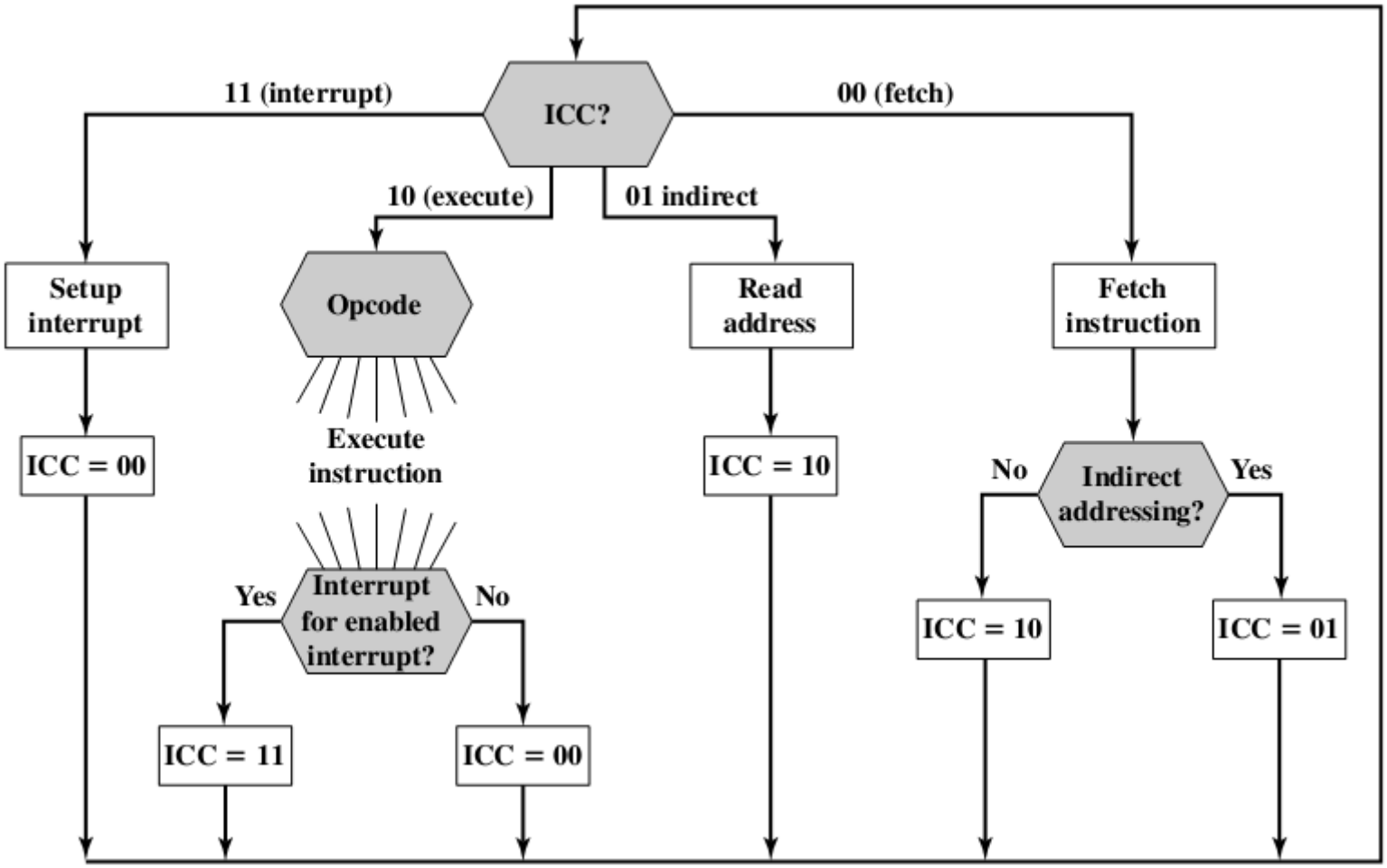
- Branch-and-save-address instruction

```
BSA X          t1: MAR ← (IR(address))
                   MBR ← (PC)
               t2: PC ← (IR(address))
                   Memory ← (MBR)
               t3: PC ← (PC) + I
```

Instruction Cycle

- There is one sequence each for the fetch, indirect, and interrupt cycles, and, for the execute cycle, there is one sequence of micro-operations for each opcode
- Assume a new 2-bit register called the instruction cycle code (ICC), which designates the state of the processor in terms of which portion of the cycle it is in
  - 00: Fetch
  - 01: Indirect
  - 10: Execute
  - 11: Interrupt

Instruction Cycle (cont.)



Functional Requirements of Processor Control

- By reducing the operation of the processor to its most fundamental level, we are able to define exactly what it is that the control unit must cause to happen, i.e., define the functional requirements for the control unit
- Three steps:
  - Define the basic elements of the processor
  - Describe the micro-operations that the processor performs
  - Determine the functions that the control unit must perform to cause the micro-operations to be performed

Functional Requirements of Processor Control (cont.)

- Basic functional elements of the processor
  - ALU
  - Registers
  - Internal data paths
  - External data paths
  - Control unit

Functional Requirements of Processor Control (cont.)

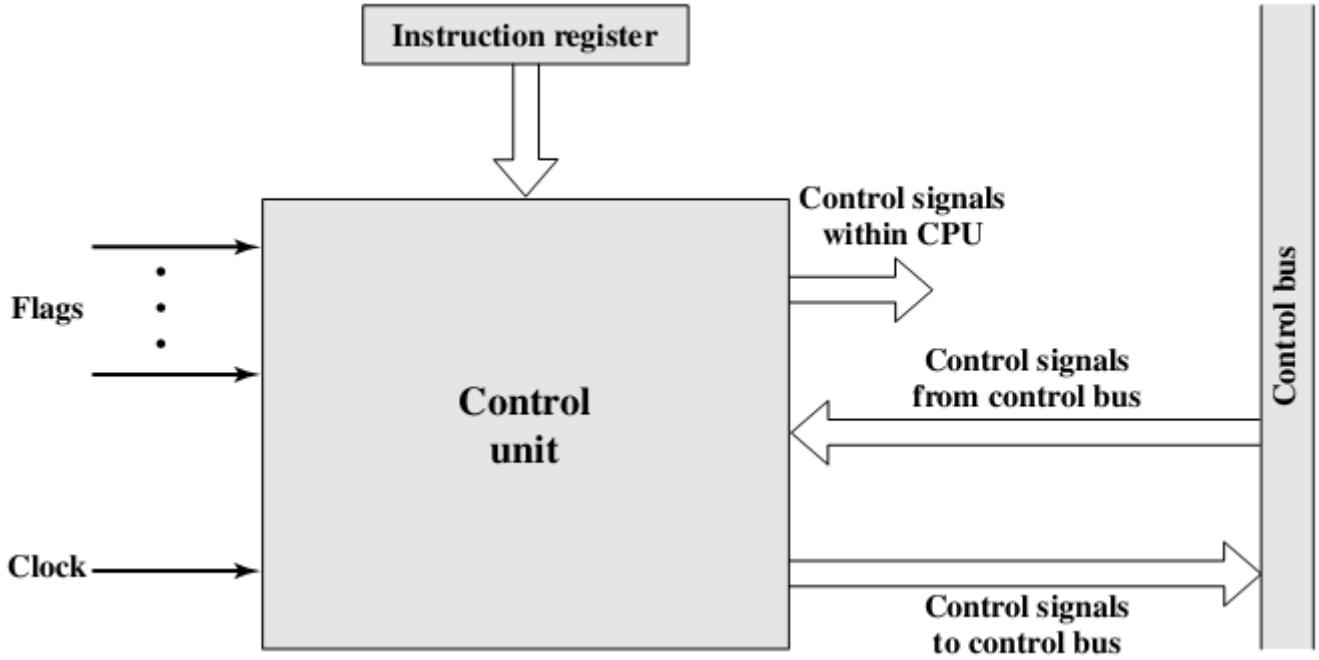
- All micro-operations fall into one of the following categories
  - Transfer data from one register to another
  - Transfer data from a register to an external interface (e.g., system bus)
  - Transfer data from an external interface to a register
  - Perform an arithmetic or logic operation, using registers for input and output

Functional Requirements of Processor Control (cont.)

- Two basic tasks of control unit
  - Sequencing: The control unit causes the processor to step through a series of micro-operations in the proper sequence, based on the program being executed
  - Execution: The control unit causes each micro-operation to be performed

Input / Output of Control Unit

- General model of control unit with all inputs and outputs



Input / Output of Control Unit (cont.)

- Input
  - Clock: control unit causes one micro-operation or a set of simultaneous micro-operations to be performed for each clock pulse
  - Instruction register: opcode and addressing mode of current instruction
  - Flags: determine the status of the processor and the outcome of previous ALU operations
  - Control signals from control bus: provide signals to the control unit, such as interrupt request and response

Input / Output of Control Unit (cont.)

- Output
  - Control signals within the processor: These are two types: those that cause data to be moved from one register to another, and those that activate specific ALU functions
  - Control signals to control bus: These are also of two types: control signals to memory, and control signals to the I/O modules

Control Signal

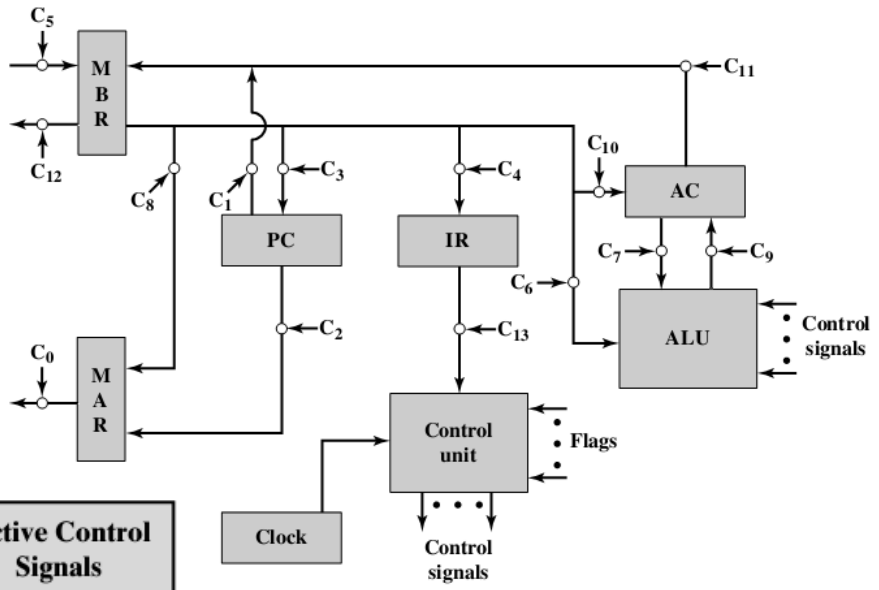
- Three types of control signals
  - Activate an ALU function
  - Activate a data path
  - Signals on the external system bus or other external interface
- All of these signals are ultimately applied directly as binary inputs to individual logic gates

Control Signal (cont.)

- Example: Fetch cycle
  - Transfer the contents of PC to MAR
    - A control signal that opens the gates between the bits of PC and MAR
  - Read a word from memory into the MBR and increment PC
    - A control signal that opens gates, allowing the contents of MAR onto the address bus
    - A memory read control signal on the control bus
    - A control signal that opens the gates, allowing the contents of the data bus to be stored in MBR
    - Control signals to logic that add 1 to PC content and store the result back to PC

Control Signal (cont.)

- Example
  - Data paths
  - ALU
  - System bus



	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

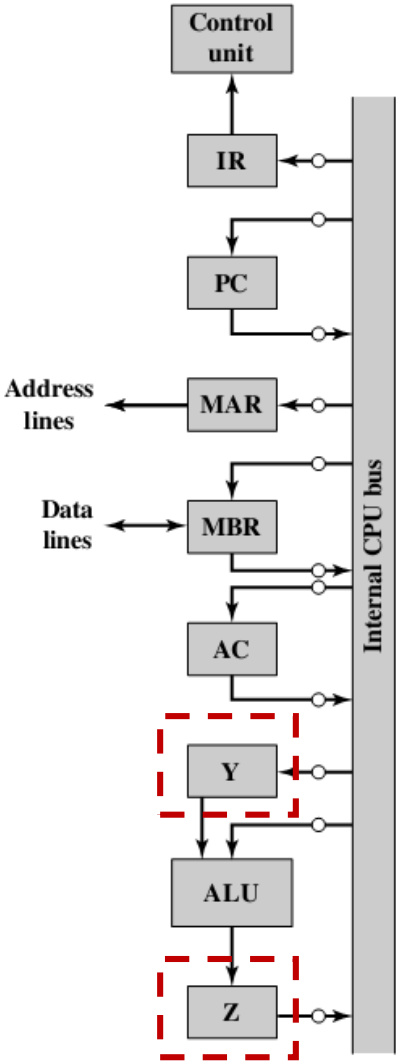
Minimal Nature of Control Unit

- It does this based only on knowing the instructions to be executed and the nature of the results of arithmetic and logical operations
  - E.g., positive, overflow, …
- It never gets to see the data being processed or the actual results produced
- It controls everything with a few control signals to points within the processor and a few control signals to the system bus

Internal Processor Bus

- A single internal bus connects the ALU and all processor registers
- Gates and control signals are provided for movement of data onto and off the bus from each register
- Additional control signals control data transfer to and from the system (external) bus and the operation of the ALU

t_1: MAR ← (IR(address))
t_2: MBR ← Memory
t_3: Y ← (MBR)
t_4: Z ← (AC) + (Y)
t_5: AC ← (Z)



Control Unit Implementation

- Two categories:
  - Hardwired implementation
  - Microprogrammed implementation
- In a hardwired implementation, the control unit is essentially a state machine circuit
  - Its input logic signals are transformed into a set of output logic signals, which are the control signals

Hardwired Implementation

- Control unit inputs
  - Key inputs: instruction register (IR), clock, flags, and control bus signals
  - In the case of the flags and control bus signals, each individual bit typically has some meaning
  - The other two inputs, IR and clock, are not directly useful to the control unit

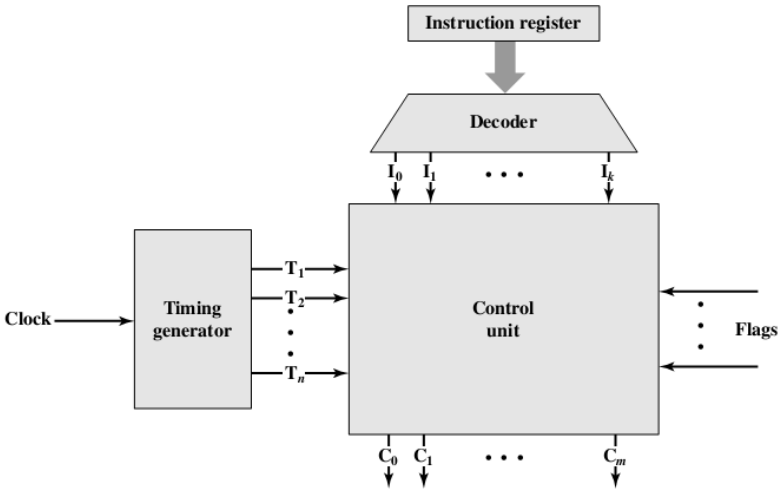
Hardwired Implementation (cont.)

- Control unit inputs: IR
  - To simplify the control unit logic, there should be a unique logic input for each opcode
  - A decoder with n binary inputs has 2^n binary outputs
  - The decoder for a control unit is more complex to account for variable-length opcodes

I1	I2	I3	I4	O1	O2	O3	O4	O5	O6	O7	O8	O9	O10	O11	O12	O13	O14	O15	O16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Hardwired Implementation (cont.)

- Control unit inputs: clock
  - The period of the clock pulses must be long enough to allow the propagation of signals along data paths and through processor circuitry
  - Control unit emits different control signals at different time units within a single instruction cycle
  - Use a counter as input to control unit, and control unit must feed back to the counter to reinitialize it



Hardwired Implementation (cont.)

- Control unit logic
  - What must be done is, for each control signal, to derive a Boolean expression of that signal as a function of the inputs
  - Define two new control signals P and Q
    - PQ = 11 Interrupt Cycle
    - PQ = 10 Execute Cycle
    - PQ = 01 Indirect Cycle
    - PQ = 00 Fetch Cycle

Hardwired Implementation (cont.)

- Control unit logic (cont.)
  - Example
    - C5: cause data to be read from the external data bus into MBR
    - C5 is asserted during the second time unit of both fetch and indirect cycles

$$C_5 = \bar{P} \cdot \bar{Q} \cdot T_2 + \bar{P} \cdot Q \cdot T_2$$

    - C5 is also needed during the execute cycle

$$C_5 = \bar{P} \cdot \bar{Q} \cdot T_2 + \bar{P} \cdot Q \cdot T_2 + P \cdot \bar{Q} \cdot (LDA + ADD + AND) \cdot T_2$$

Microprogramming Language

- Microinstruction: Each line describes a set of micro-operations occurring at one time
- Microprogram / firmware: A sequence of instructions
- Microprogram is midway between hardware and software
  - It is easier to design in firmware than hardware, but it is more difficult to write a firmware program than a software program

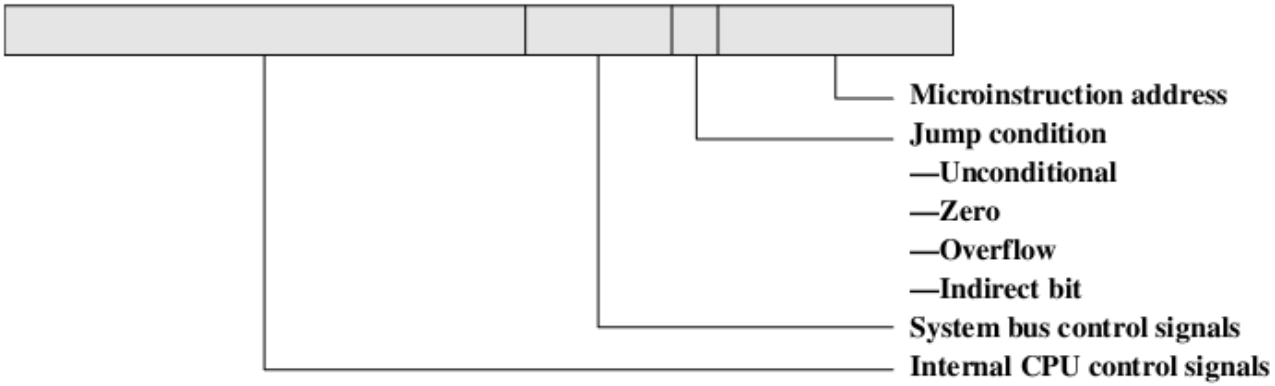
	Micro-operations	Active Control Signals
Fetch:	t_1: MAR ← (PC)	C_2
	t_2: MBR ← Memory	C_5, C_R
	PC ← (PC) + 1	
	t_3: IR ← (MBR)	C_4
Indirect:	t_1: MAR ← (IR(Address))	C_8
	t_2: MBR ← Memory	C_5, C_R
	t_3: IR(Address) ← (MBR(Address))	C_4
Interrupt:	t_1: MBR ← (PC)	C_1
	t_2: MAR ← Save-address	
	PC ←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

Basic Idea

- Consider that for each micro-operation, all that the control unit is allowed to do is generate a set of control signals, to control each control line emanating from the control unit is either on or off
- Each control line is represented by a binary digit
- A control word is constructed in which each bit represents one control line
- Each micro-operation would be represented by a different pattern of 1s and 0s in the control word

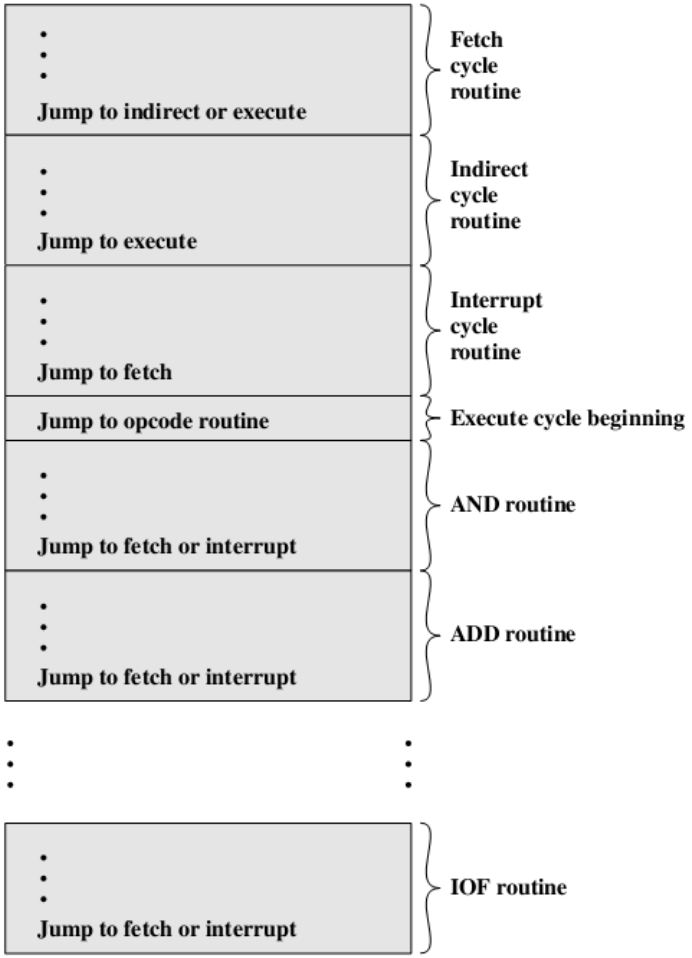
Basic Idea (cont.)

- The sequence of micro-operations is represented by stringing together a sequence of control words
- For the sequence of micro-operations is not fixed, control words is put in a memory, with each word having a unique address
  - Add an address field to each control word, indicating the location of next control word to be executed if a certain condition is true
  - Add a few bits to specify the condition



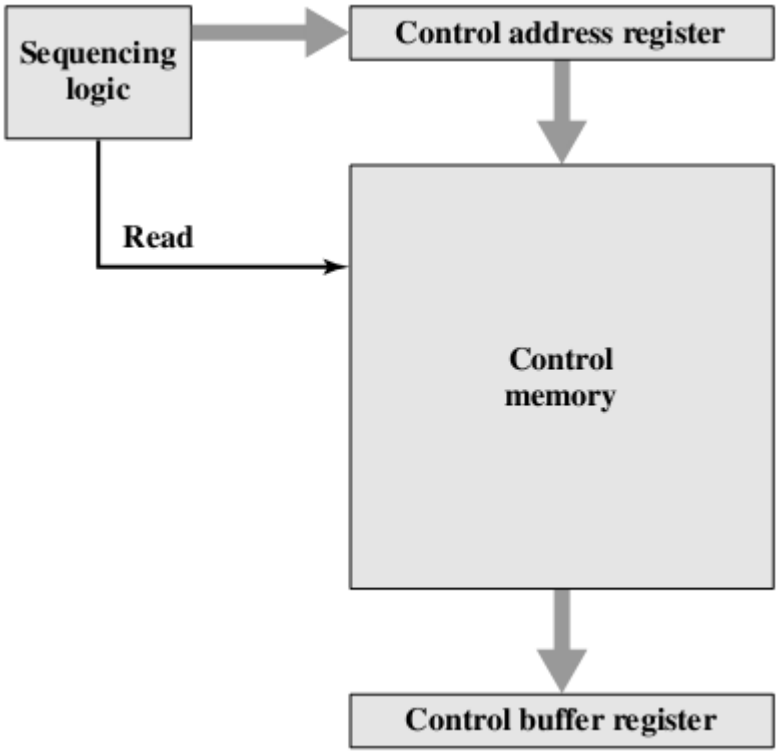
Microinstruction Interpretation

- To execute this microinstruction, turn on all the control lines indicated by a 1 bit; leave off all control lines indicated by a 0 bit
- If the condition indicated by the condition bits is false, execute the next microinstruction in sequence
- If the condition indicated by the condition bits is true, the next microinstruction to be executed is indicated in the address field



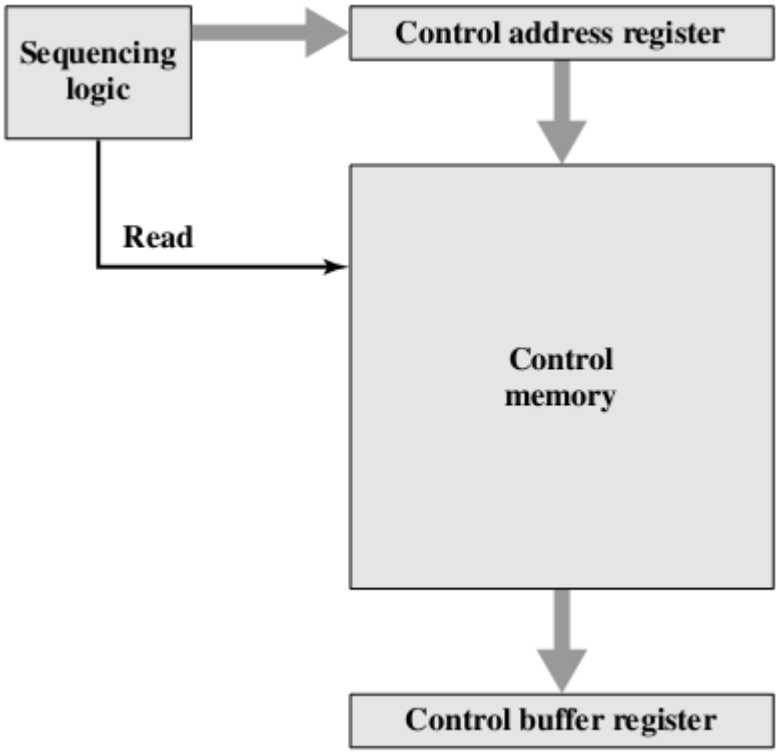
Microprogrammed Control Unit

- Control memory: Store microinstructions
- Control address register: Contain the address of the next microinstruction to be read
- Sequencing unit: Load the control address register and issue a read command



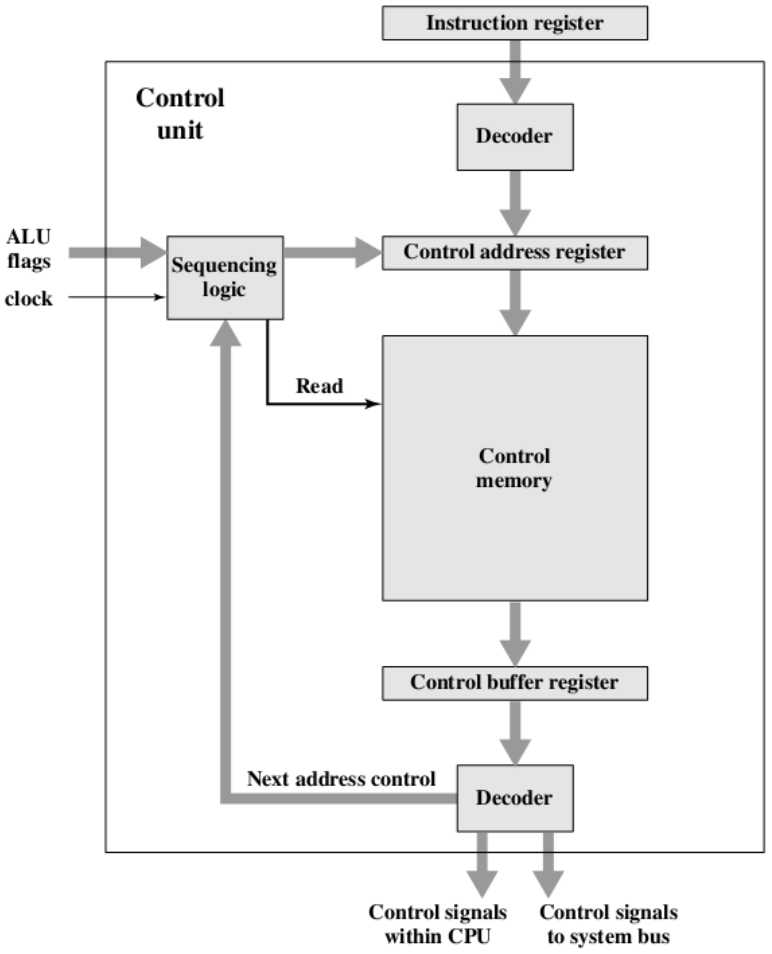
Microprogrammed Control Unit (cont.)

- When a microinstruction is read from the control memory, it is transferred to a control buffer register
- The left-hand portion of that register connects to the control lines emanating from the control unit
- Reading a microinstruction from the control memory is the same as executing that microinstruction



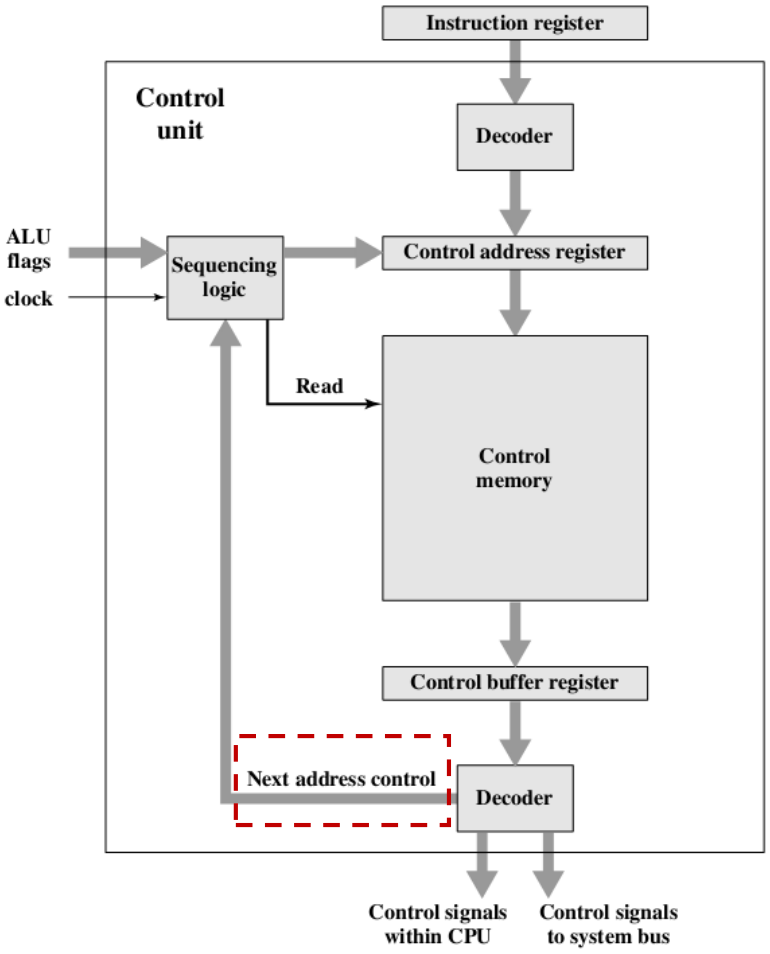
Microprogrammed Control Unit (cont.)

- Sequencing logic unit issues a READ command to the control memory
- The word whose address is specified in control address register is read into control buffer register
- The content of control buffer register generates control signals and next-address information for sequencing logic unit
- The sequencing logic unit loads a new address into control address register based on the next-address information from control buffer register and ALU flags



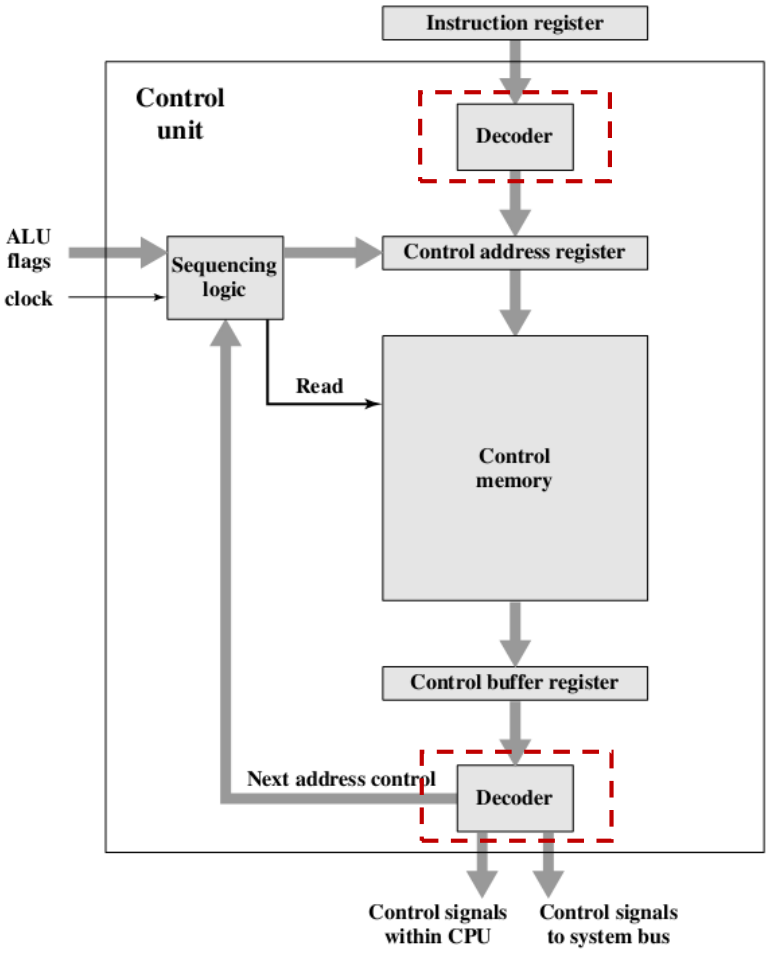
Microprogrammed Control Unit (cont.)

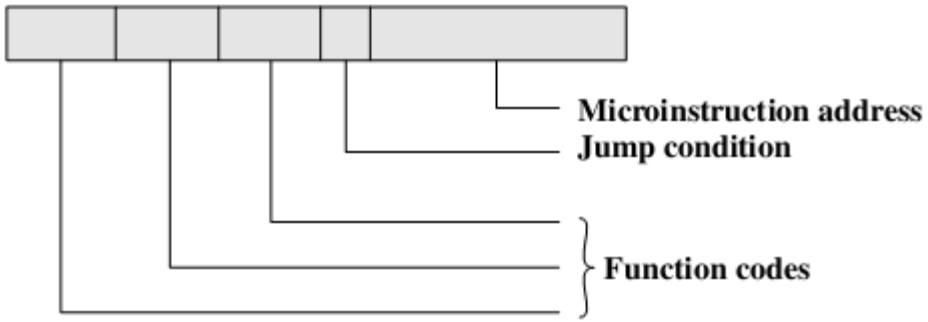
- Three choices of the last step:
  - Get the next instruction: Add 1 to the control address register
  - Jump to a new routine based on a jump microinstruction: Load the address field of the control buffer register into the control address register
  - Jump to a machine instruction routine: Load the control address register based on the opcode in the IR



Microprogrammed Control Unit (cont.)

- Upper decoder: translate the opcode of IR into a control memory address
- Lower decoder: not used for horizontal microinstructions but is used for vertical microinstructions





Advantages and Disadvantages

- Advantages
  - Simplify the design of the control unit
    - It is both cheaper and less error prone to implement
- Disadvantages
  - somewhat slower than a hardwired unit of comparable technology

Task of Microprogrammed Control Unit

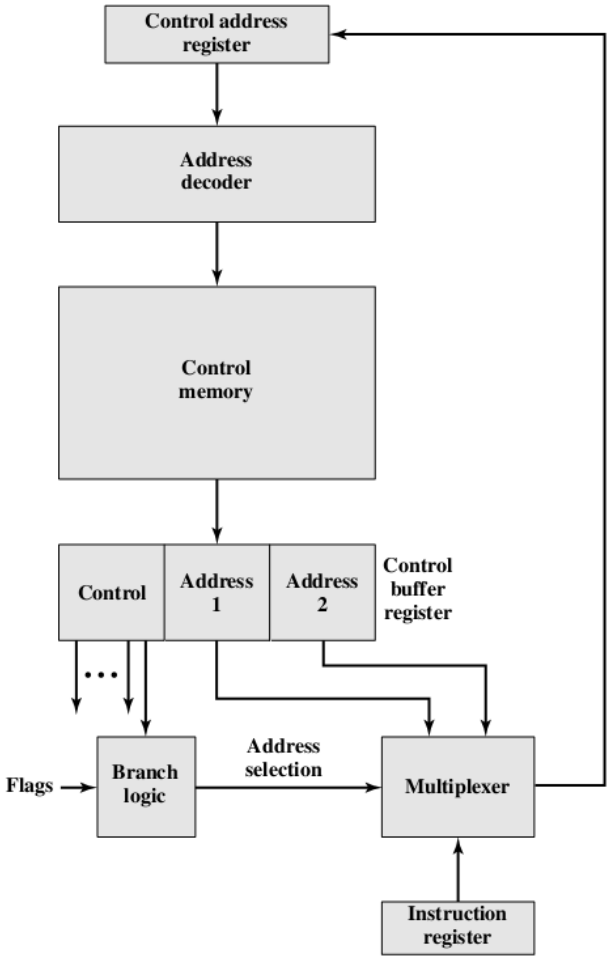
- Microinstruction sequencing: Get the next microinstruction from the control memory
- Microinstruction execution: Generate the control signals needed to execute the microinstruction

Microinstruction Sequencing

- Design considerations
  - Size of microinstruction: minimizing the size of the control memory reduces the cost of that component
  - Address-generation time: a desire to execute microinstructions as fast as possible
- Sequencing technique
  - Based on the current microinstruction, condition flags, and the contents of the instruction register, a control memory address must be generated for the next microinstruction

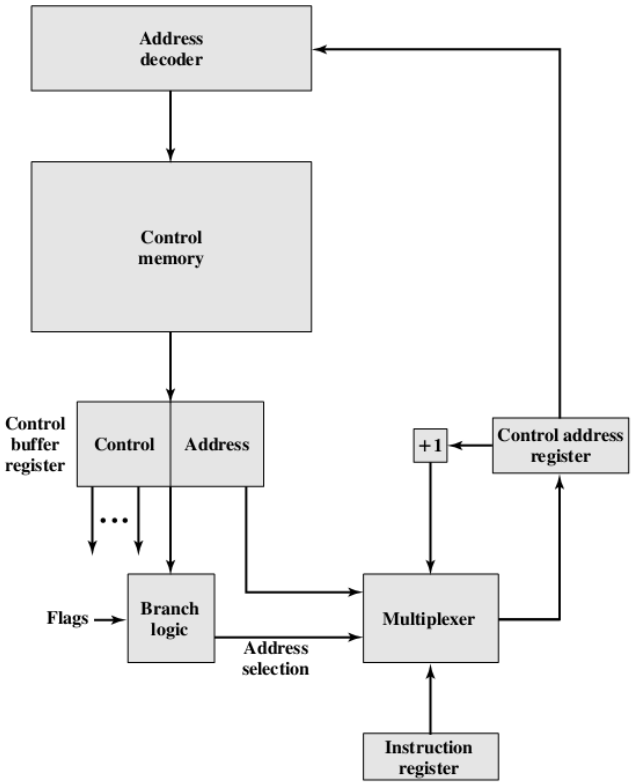
Microinstruction Sequencing (cont.)

- Category
  - Two address fields
    - Provide two address fields in each microinstruction
    - Based on an address-selection input, the multiplexer transmits either the opcode or one of the two addresses to the control address register



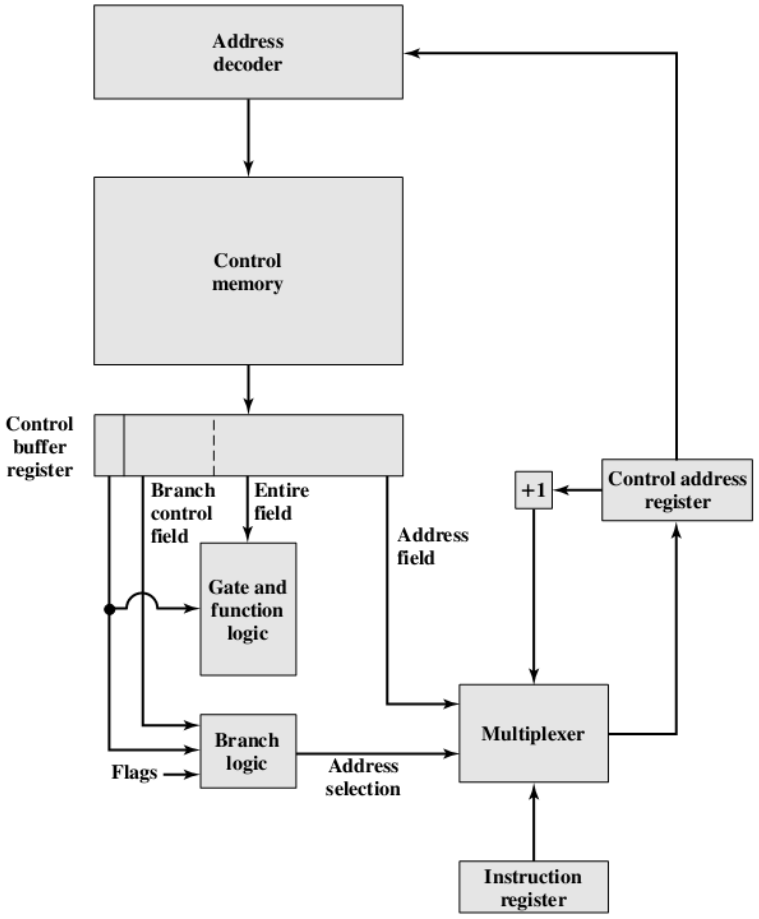
Microinstruction Sequencing (cont.)

- Category (cont.)
  - Single address field
    - The address-selection signals determine which option is selected



Microinstruction Sequencing (cont.)

- Category
  - Variable format
    - Provide two entirely different microinstruction formats, and use one bit to designate which format is being used



Microinstruction Sequencing (cont.)

- Address generation
  - Explicit
    - Two-field
    - Unconditional branch
    - Conditional branch
  - Implicit
    - Mapping
    - Addition
    - Residual control

Microinstruction Execution

- The effect of the execution of a microinstruction is to generate control signals
  - Some of these signals control points internal to the processor
  - The remaining signals go to the external control bus or other external interface
- As an incidental function, the address of the next microinstruction is determined

Summary

- Micro-operations
  - Concept, different cycles
- Functional requirements of processor control
- Input / output of control unit
- Control signal
- Hardwired implementation

Summary

- Microprogramming language
  - Microinstruction, microprogram
- Microinstruction interpretation
- Microprogrammed control unit
  - Element, execution, procedure, advantages and disadvantages
- Task of microprogrammed control unit
  - Microinstruction sequencing, microinstruction execution

Thank You

rentw@nju.edu.en

南京大學
NANJING UNIVERSITY