Computer Organization and Architecture

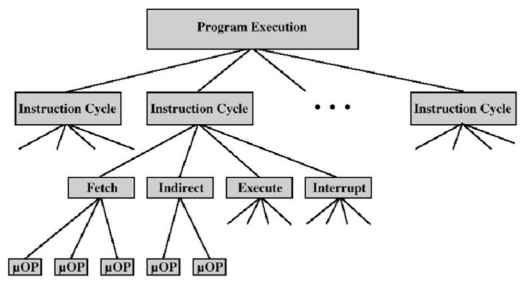
Chapter 15

Control Unit Operation

Micro-Operations

- Execution of an instruction (the instruction cycle) has a number of smaller units
  - Fetch, indirect, execute, interrupt, etc
- Each part of the cycle has a number of smaller steps called micro-operations
  - Discussed extensive in pipelining
- Micro-ops are the fundamental or atomic operations of the processor

Constituents of Program Execution



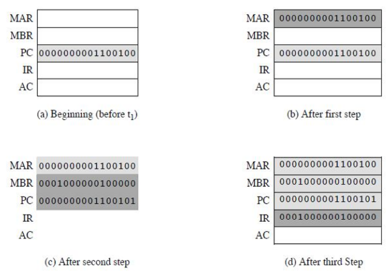
The Fetch Cycle: 4 Registers

- Memory Address Register (MAR)
  - Connected to address bus
  - Specifies address for read or write op
- Memory Buffer Register (MBR)
  - Connected to data bus
  - Holds data to write or last data read
- Program Counter (PC)
  - Holds address of next instruction to be fetched
- Instruction Register (IR)
  - Holds last instruction fetched

Fetch Sequence

- Address of next instruction is in PC
  - Address (MAR) is placed on address bus
  - Control unit issues READ command
- Result (data from memory) appears on data bus
  - Data from data bus copied into MBR
  - PC incremented by instruction length (in parallel with data fetch from memory)
- Data (instruction) moved from MBR to IR
  - MBR is now free for further data fetches

Fetch example



Fetch Sequence (symbolic)

- t1: MAR <- (PC)
- t2: MBR <- (memory)
- PC <- (PC) +1
- t3: IR <- (MBR)
- (tx = time unit/clock cycle)
- or
- t1: MAR <- (PC)
- t2: MBR <- (memory)
- t3: PC <- (PC) +1
- IR <- (MBR)

Fetch Sequence - Symbolic

- The fetch cycle actually consists of 3 step and 4 micro-ops
- Each micro-op consists of moving data in or out of a register
- Those that do not conflict can be executed in parallel

```
    t1:     MAR <- (PC)
    t2:     MBR <- (memory)
            PC <- (PC) +1
    t3:     IR <- (MBR)
or
    t1:     MAR <- (PC)
    t2:     MBR <- (memory)
    t3:     PC <- (PC) +1
            IR <- (MBR)
```

Rules for Grouping Micro-ops

- Proper sequence must be followed
  - MAR <- (PC) must precede MBR <- (memory)
- Conflicts must be avoided
  - Must not read & write same register in same cycle
  - MBR <- (memory) & IR <- (MBR) must not be in same cycle
- Also PC <- (PC) +1 involves addition
  - Might need to Use ALU
  - May need additional micro-operations

Indirect Cycle

- Once the instruction has been fetched we need to fetch source operands.
  - Assume one-address instruction format with direct and indirect addressing allowed
  - Indirect cycle (memory at addr contains address of operand):

    t1: MAR <- ($IR_{address}$) - address field of IR

    t2: MBR <- (memory)

    t3: $IR_{address}$ <- ($MBR_{address}$)

  Now MBR contains direct address of operand
- IR is updated with direct address of operand
- IR is now in same state as if direct addressing had been used

Interrupt Cycle

- At end of execute cycle, processor tests interrupt signal. If set, an interrupt cycle occurs

```
    t1:     MBR <-(PC)
    t2:     MAR <- save-address
            PC <- routine-address
    t3:     memory <- (MBR)
```

- This is a minimum. Most processors provide multiple types of address
  - So there may be additional micro-ops to get addresses
  - Note that saving context is done by interrupt handler routine, not micro-ops

Execute Cycle (ADD)

- Fetch, Indirect and Interrupt cycles are simple and predictable
- Execute cycle is different for each instruction
- We'll look at several examples
- ADD R1,X - add the contents of location X to Register 1 , result in R1

  t1: MAR <- ($IR_{address}$)

  t2: MBR <- (memory)

  t3: R1 <- R1 + (MBR)
- Example is simplified. We may need additional micro-ops to get register reference from IR or stage ALU input or output in an intermediate register

Execute Cycle (ISZ)

- ISZ X - increment and skip if zero
- Contents of location X are incremented by 1; if the result is 0 the next instruction is skipped
  - t1: MAR <- ($IR_{address}$)
  - t2: MBR <- (memory)
  - t3: MBR <- (MBR) + 1
  - t4: memory <- (MBR)
  - if (MBR) == 0 then PC <- (PC) + 1
- Notes:
  - Conditional action (test and increment if 0) is a single micro-op
  - Can be performed in same time unit as store to memory

Execute Cycle (BSA)

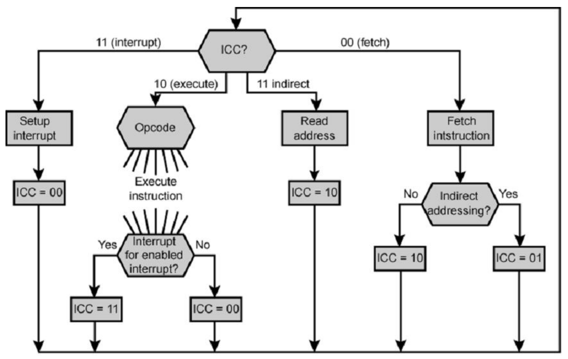
- BSA X - Branch and save address
  - Address of instruction following BSA is saved in X
  - Execution continues from X+1
  - A straightforward way to provide subroutine calls
  - But you can get into trouble with recursive calls; most modern machines use a stack

```
t1:   MAR <- (IR_address)
      MBR <- (PC)
t2:   PC <- (IR_address)
      memory <- (MBR)
t3:   PC <- (PC) + 1
```

Instruction Cycle

- Each phase is decomposed into a sequence of elementary micro-ops
- We have one sequence for fetch, indirect, and interrupt cycles, but execute cycle has one sequence of micro-operations for each opcode
- To complete the picture we need to tie sequences together into the instruction cycle
  - Assume new 2-bit register; the instruction cycle code (ICC) designates which part of cycle processor is in
    - 00: Fetch
    - 01: Indirect
    - 10: Execute
    - 11: Interrupt

Flowchart for Instruction Cycle



Control of the Processor

- Functional Requirements
  - Define the basic elements of the processor
  - Describe the micro-operations that the processor performs
  - Determine the functions control unit must perform in order to execute the micro-ops
- We've already completed steps 1 and 3

Basic Elements of Processor

- Define the basic elements of the processor:
  - ALU
  - Registers
  - Internal data pahs
  - External data paths
  - Control Unit

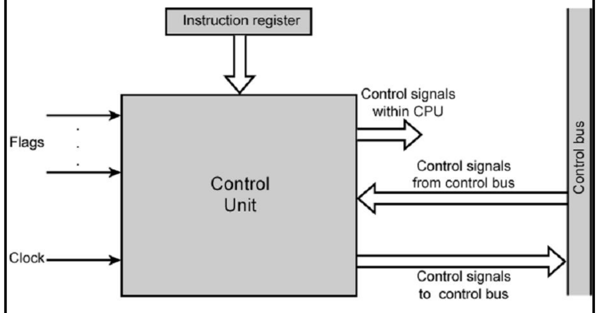
Types of Micro-operation

- Describe the micro-operations that the processor performs
  - Transfer data between registers
  - Transfer data from register to external interface
  - Transfer data from external interface to register
  - Perform arithmetic or logical operations using registers

Functions of Control Unit

- Control Unit performs two basic tasks
- Sequencing
  - Causing the CPU to step through a series of micro-operations
- Execution
  - Causing the performance of each micro-op
- Key to operation is the use of control signals

Model of Control Unit


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

Control Unit Inputs

- Clock
  - One micro-op (or set of parallel micro-ops) per clock cycle
- Instruction register
  - Contains op-code for current instruction
  - Determines which micro-ops are performed
- Flags
  - Determine state of CPU and results of previous operations
- Control signals from control bus
  - The control lines from the system bus are input signals to the control unit

Control Unit Outputs

- Control signals within the processor
  - Cause data movement from reg to reg
  - Activate specific ALU functions
- To control bus
  - Control signals to memory
  - Control signals t
  - o I/O modules

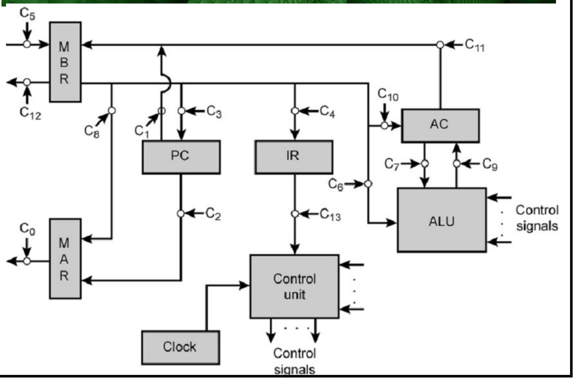
Example Control Signal Sequence - Fetch

- MAR <- (PC)
  - Control unit activates signal to open gates between PC and MAR
- MBR <- (memory) uses these simultaneous signals
  - Open gates between MAR and address bus (places MAR on address bus)
  - Memory read control signal is sent on the control bus
  - Open gates between data bus and MBR, allowing contents of data bus to be stored in MBR
  - Control signals to PC increment logic circuit
- After this is complete the control unit examines IR to determine whether to perform an indirect cycle or an execute cycle

Control Signals Example

- This model is a simple processor that has one register AC (accumulator)
- Diagram indicates data paths between elements
  - Terminations of control signals are labeled C_n and indicated by a circle
  - Inputs are clock, flags, IR
- With each clock cycle the control unit reads all of its inputs and emits a set of control signals

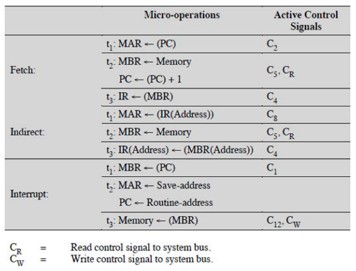
Data Paths and Control Signals



Control Signal Destinations

- Data paths
  - Switching & connecting data sources to destinations; e.g., connect IR to MBR on instruction fetch
- ALU
  - Signals activate various logic circuits in ALU
- System Bus
  - Control signals such as memory read or write

Control Signals for Fetch, Interrupt, Indirect



	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

C_R = Read control signal to system bus.
C_W = Write control signal to system bus.

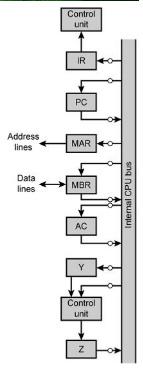
Internal Organization

- Control signals diagram shows a variety of data paths in a very simple processor
  - Complexity would be to high in any real processor to have hardwired data paths
  - Usually a single internal bus is used
  - Gates control movement of data onto and off the bus
- Control signals control data transfer to and from external systems bus

Alternate Organization with Internal Bus

- Two new registers Y and Z have been added for proper operation of ALU
- Y is the input and Z the output
  - The ALU is a combinatorial circuit with no internal storage
  - When control signals are applied the input is transformed to the output
  - It is not connected directly to the bus because the output would feed back to the input
- Micro-ops for Add to Memory

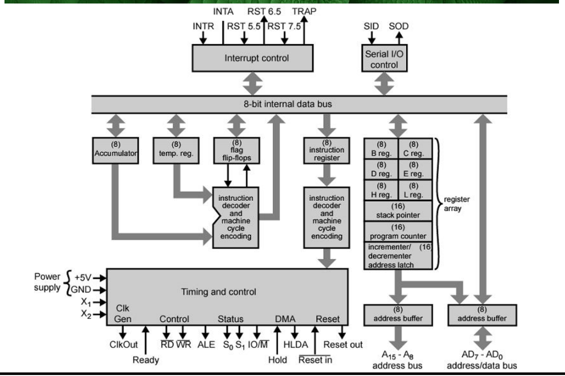
T1: MAR <- (IR(address))
T2: MBR <- Memory
T3: Y <- (MBR)
T4: Z <- (AC) + (Y)
T5: AC <- (Z)



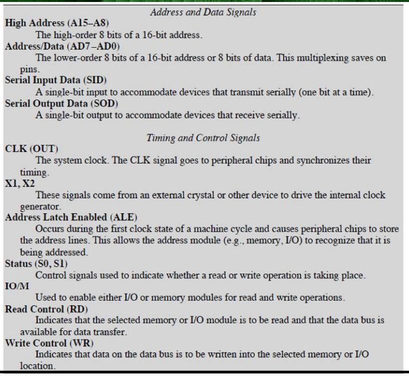
The Intel 8085

- An 8-bit microprocessor produced in 1977
- Some key components that may not be obvious:
  - Incrementer/Decrementer address latch: add or subtract 1 from SP or PC. Saves time by not using ALU
  - Interrupt Control: handles multiple levels or interrupt signals
  - Serial I/O control: interface for serial devices (1 bit at a time)

Intel 8085 CPU Block Diagram



8085 External Signals - 1



Address and Data Signals

High Address (A15–A8)
The high-order 8 bits of a 16-bit address.

Address/Data (AD7–AD0)
The lower-order 8 bits of a 16-bit address or 8 bits of data. This multiplexing saves on pins.

Serial Input Data (SID)
A single-bit input to accommodate devices that transmit serially (one bit at a time).

Serial Output Data (SOD)
A single-bit output to accommodate devices that receive serially.

Timing and Control Signals

CLK (OUT)
The system clock. The CLK signal goes to peripheral chips and synchronizes their timing.

X1, X2
These signals come from an external crystal or other device to drive the internal clock generator.

Address Latch Enabled (ALE)
Occurs during the first clock state of a machine cycle and causes peripheral chips to store the address lines. This allows the address module (e.g., memory, I/O) to recognize that it is being addressed.

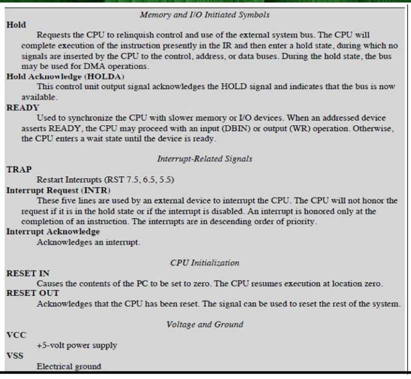
Status (S0, S1)
Control signals used to indicate whether a read or write operation is taking place.

IO/M
Used to enable either I/O or memory modules for read and write operations.

Read Control (RD)
Indicates that the selected memory or I/O module is to be read and that the data bus is available for data transfer.

Write Control (WR)
Indicates that data on the data bus is to be written into the selected memory or I/O location.

8085 External Signals - 2



Memory and I/O Initiated Symbols

Hold
Requests the CPU to relinquish control and use of the external system bus. The CPU will complete execution of the instruction presently in the IR and then enter a hold state, during which no signals are inserted by the CPU to the control, address, or data buses. During the hold state, the bus may be used for DMA operations.

Hold Acknowledge (HOLDA)
This control unit output signal acknowledges the HOLD signal and indicates that the bus is now available.

READY
Used to synchronize the CPU with slower memory or I/O devices. When an addressed device asserts READY, the CPU may proceed with an input (DBIN) or output (WR) operation. Otherwise, the CPU enters a wait state until the device is ready.

Interrupt-Related Signals

TRAP
Restart Interrupts (RST 7.5, 6.5, 5.5)

Interrupt Request (INTR)
These five lines are used by an external device to interrupt the CPU. The CPU will not honor the request if it is in the hold state or if the interrupt is disabled. An interrupt is honored only at the completion of an instruction. The interrupts are in descending order of priority.

Interrupt Acknowledge
Acknowledges an interrupt.

CPU Initialization

RESET IN
Causes the contents of the PC to be set to zero. The CPU resumes execution at location zero.

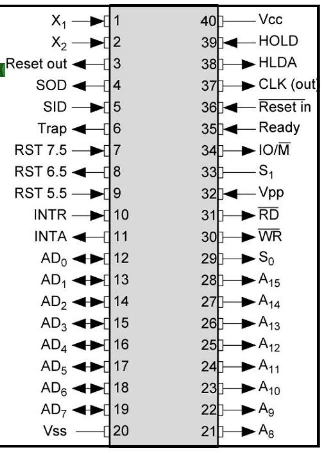
RESET OUT
Acknowledges that the CPU has been reset. The signal can be used to reset the rest of the system.

Voltage and Ground

VCC
+5-volt power supply

VSS
Electrical ground

Intel 8085 Pin Configuration



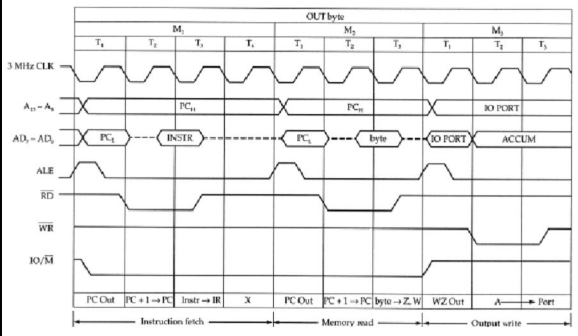
Control Unit

- Two components
  - Instruction decoder / machine cycle encoding
  - Timing and control
- Essence is Timing and Control
  - Inputs are clock, current instruction and some control signals
  - Outputs are control signals to processor and external bus

Instruction Timing

- Timing is synchronized by the clock
  - Instruction cycle is divided into 1 to 5 machine cycles depending on instruction
  - Each machine cycle is divided into 3 to 5 states
  - Each state lasts one clock cycle
  - Processor performs one or more parallel micro-ops per state (determined by control signals)
  - Machine cycles are defined to equivalent to bus accesses
    - Determined by number of processor communicates with external devices
    - Ex: 16-bit read mem instruction needs two cycles to fetch instruction plus one cycle for execution
    - Compare with modern processors where bus is much slower than CPU

Intel 8085 OUT Instruction Timing Diagram



External Control Signals Examples

- Instruction is OUT byte (output to IO device); 3 machine cycles
  1. Instruction opcode is fetched
  2. 2nd half of instruction is fetched with I/O address
  3. Contents of AC written out to device over data bus

Machine Cycle Detail M1

- Address Line Enable (ALE) signal start of each machine cycle; alerts external circuits
- In state t1 of m1 control unit set IO/M to indicate memory op
  - Contents of PC placed on address bus and address/data bus
  - With falling edge of ALE other devices latch (store) the addr
- Timing state T2 the memory module places contents of memory location on addr/data bus
  - Control unit sets RD signal to indicate a read but waits until T3 to copy the data
  - Gives memory module time to put the data on the bus and stabilize signal levels
- State T3 is bus idle state during which processor decodes the instruction

Implementation

- Control unit design techniques are either
  - hardwired
  - Or microprogrammed
- In a hardwired implementation the control unit is a state machine
- Input logic signals are transformed into output signals (control signals)

Hardwired Implementation (1)

- Control unit inputs:
  - Flags and control bus
    - Each bit means something
  - Instruction register not directly useful to the unit
    - Op-code causes different control signals for each different instruction
    - Unique logic for each op-code
    - Decoder takes encoded input and produces single output
    - *n* binary inputs and 2^n outputs
    - Each of the 2^n input patterns will produce a unique output
  - Example shows a very simple 4-bit decoder

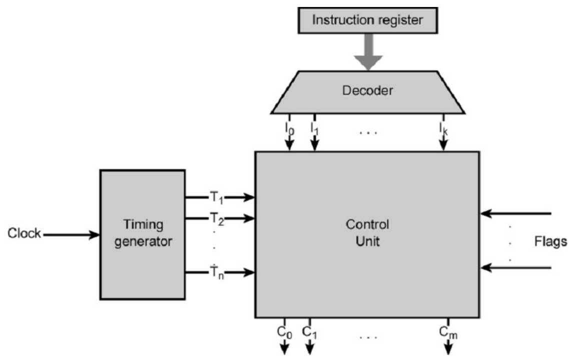
Decoder with 4 inputs and 16 outputs

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 (2)

- Clock
  - — Repetitive sequence of pulses
  - — Useful for measuring duration of micro-ops
  - — Must be long enough to allow signal propagation
  - — Different control signals at different times within instruction cycle
  - — Need a counter with different control signals for t1, t2 etc.

Control Unit with Decoded Inputs



Operation of Control Unit

- For each control signal, derive a Boolean expression of signal as function of inputs
- Consider logical view of control unit data paths discussed earlier
- Look at control signal C5
  - — Causes data to be read from external device into MBR
  - — Used by fetch T2 and indirect T2; sometimes by Execute
- We define two new control signals P and Q that are interpreted as

PQ = 00	Fetch cycle
PQ = 01	Indirect cycle
PQ = 10	Execute cycle
PQ = 11	Interrupt cycle

Boolean expression for C5

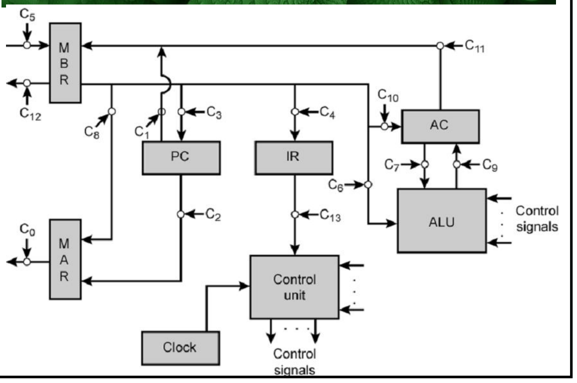
- The following expression defines C5 for fetch and indirect

  C5 = (~P ^ ~Q ^ T2) | (~P ^ Q ^ T2)
- For execute, we need a control signal for each instruction
- Assume we have three instructions that read from memory (LDA, ADD, AND)

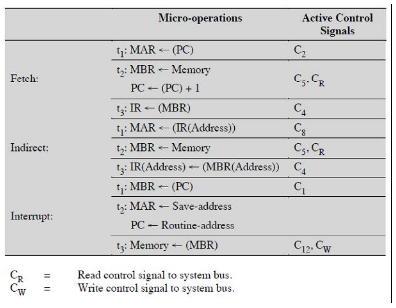
  C5 = (~P^~Q^T2) | (~P^Q^T2)

  | (P^~Q^(LDA | ADD | AND)^T2)
- Repeat this process for each control signal
- Result is a set of Boolean equations that define the behavior of the control and therefore the processor

Data Paths and Control Signals



Usage of control signals



	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

C_R = Read control signal to system bus.
C_W = Write control signal to system bus.

Problems With Hard Wired Designs

- Complex sequencing & micro-operation logic
- Difficult to design and test
  - With modern processors the number of Boolean equations is huge and implementation of combinatorial circuit is very difficult
- Inflexible design
  - Difficult to add new instructions
- Microprogramming offers a simpler and easier approach