# INTRODUCTION

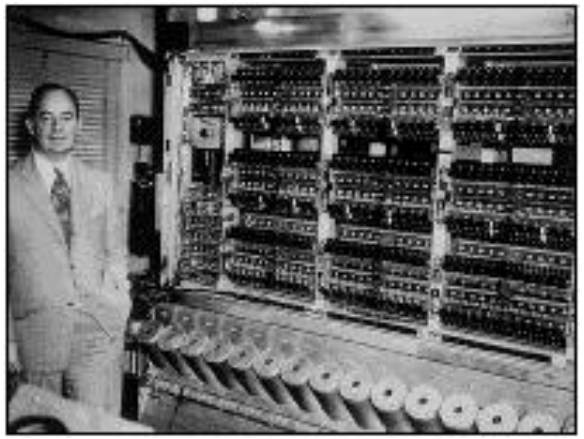
# Outline

- Review Basic Computer of System Architecture I
- Introduce Pico-processor
- Memory Access Cycle
- Input/Ouput Instructions

# Introduction: the Pico-Processor

- Objective
  - Simple design of a processor
  - Executes a few simple instructions
  - Gain understanding of processor hardware principles
- Basic idea
  - Processor is basically a collection of digital logic components

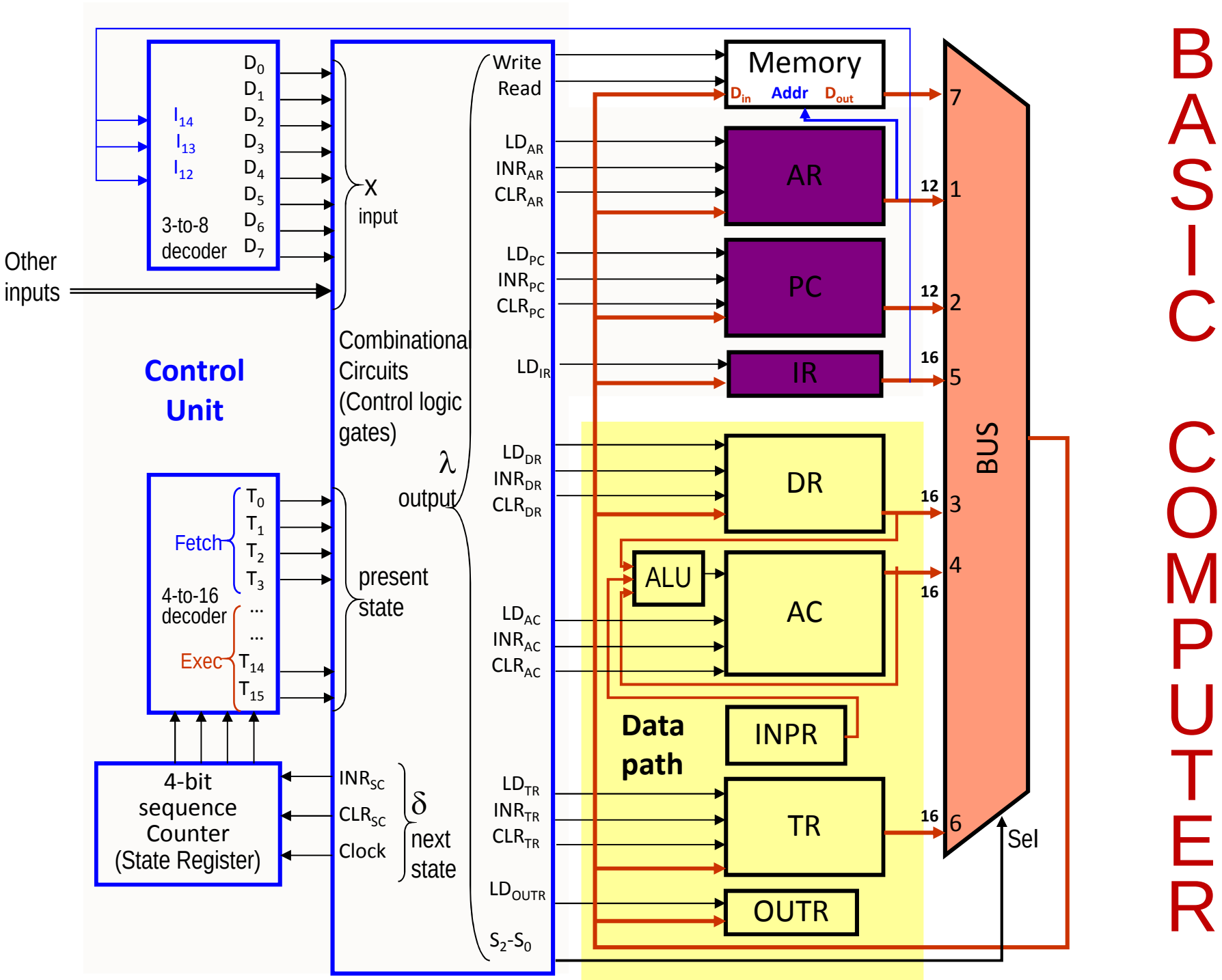
# Basic Computer Architecture



*Based on von Neumann architecture Developed in 1945*

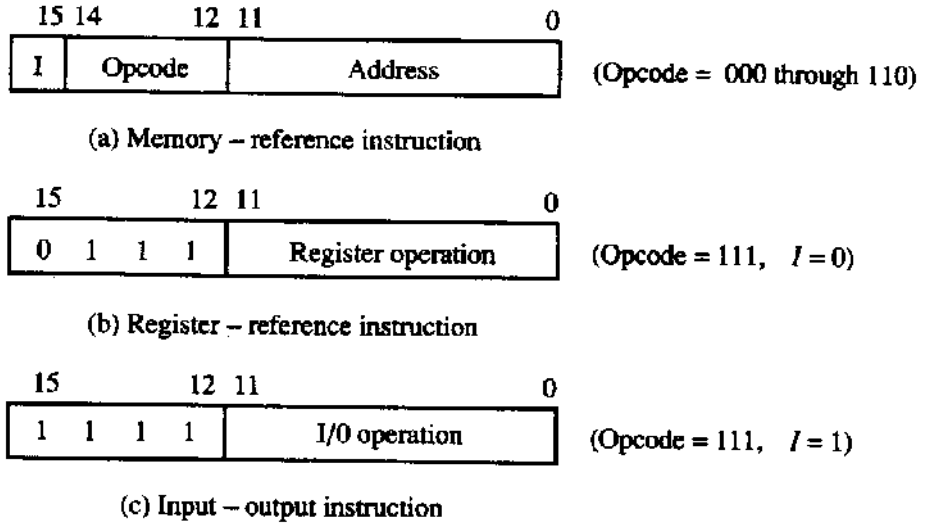
# Computer Architecture I – Reminder

- Basic Computer
  - 8 General purpose Register
    - ✓Support Load, Reset, Increment
  - 16 bit common data bus
  - 12 bit address bus
  - $2^{12}$ = 4096 word memory space = 4096 x 16 bit
  - Complete instruction set architecture
    - ✓Data manipulation
    - ✓Program flow
    - ✓Input/output


BASIC COMPUTER
Memory
Din Addr Dout
AR
PC
IR
DR
ALU
AC
INPR
TR
OUTR
BUS
Sel
Write
Read
LDAR INRAR CLRAR
LDPC INRPC CLRPC
LDIR
LDDR INRDR CLRDR
LDAC INRAC CLRAC
LDTR INRTR CLRTR
LDOUTR
S2-S0
Control Unit
Combinational Circuits (Control logic gates)
λ output
Other inputs
I14 I13 I12
3-to-8 decoder
D0 D1 D2 D3 D4 D5 D6 D7
x input
Fetch
4-to-16 decoder
Exec
T0 T1 T2 T3 ... ... T14 T15
present state
4-bit sequence Counter (State Register)
INRSC CLRSC Clock
δ next state
Data path
7 12 1 12 2 16 5 16 3 16 4 16 6

# Basic Computer Instruction Set

- There are three types of instructions which have different formats:
- Memory Reference Instruction – MRI
  - Direct Addressing (I=0)
  - Indirect Addressing (I=1)
- Register Reference Instruction - RRI
- Input-Output Instruction - IOI



| Symbol | I=0 | I= 1 | Description |
|---|---|---|---|
| **MRI** | | | |
| AND | 0xxx | 8xxx | AND memory word to AC |
| ADD | 1xxx | 9xxx | Add memory word to AC |
| LDA | 2xxx | Axxx | Load memory word to AC |
| STA | 3xxx | Bxxx | Store content of AC in memory |
| BUN | 4xxx | Cxxx | Branch unconditionally |
| BSA | 5xxx | Dxxx | Branch and save return address |
| ISZ | 6xxx | Exxx | Increment and skip if zero |
| **RRI** | | | |
| CLA | 7800 | | Clear AC |
| CLE | 7400 | | Clear E |
| CMA | 7200 | | Complement AC |
| CME | 7100 | | Complement E |
| CIR | 7080 | | Circulate right AC and E |
| CIL | 7040 | | Circulate left AC and E |
| INC | 7020 | | Increment AC |
| SPA | 7010 | | Skip next instruction if AC positive |
| SNA | 7008 | | Skip next instruction if AC negative |
| SZA | 7004 | | Skip next instruction if AC zero |
| SZE | 7002 | | Skip next instruction if E is 0 |
| HLT | 7001 | | Halt computer |
| **IOI** | | | |
| INP | | F800 | Input character to AC |
| OUT | | F400 | Output character from AC |
| SKI | | F200 | Skip on input flag |
| SKO | | F100 | Skip on output flag |
| ION | | F080 | Interrupt on |
| IOP | | F040 | Interrupt off |

BASIC COMPUTER

# Instruction Set Completeness

- A computer instruction set is complete if it can be used to evaluate any function that is known to be computable.
- To be complete, the instruction set must contain enough instructions in each of the following categories:
  1. Arithmetic, logic, and shift instructions.
  2. Instructions for moving information to and from memory and processor registers. (load/store)
  3. Program control instructions together with instructions that check status conditions.
  4. Input and output instructions

# Instruction Set Completeness

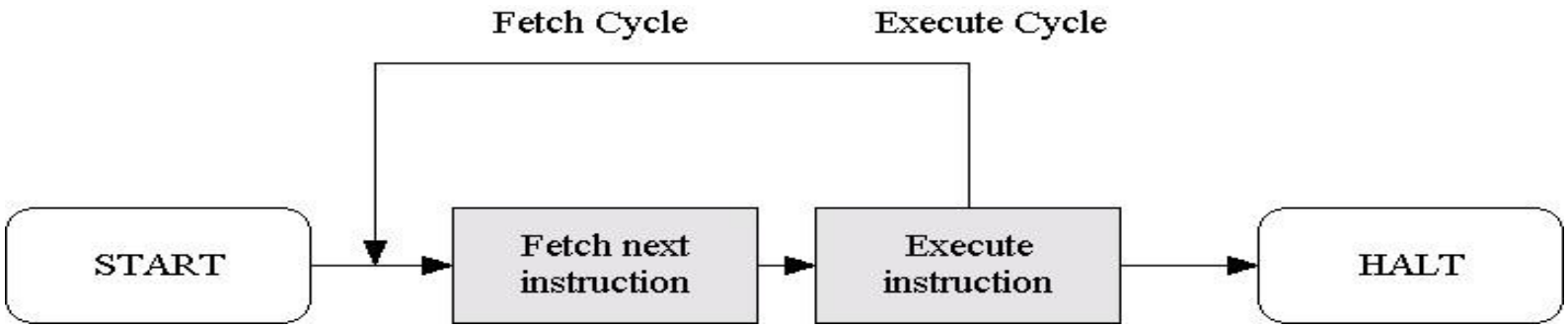
- Program control instructions, such as branch instructions, are used to change the sequence in which the program is executed.
- Input and output instructions are needed for communication between the computer and the user (outside world).
- In this context, the instruction list of the Basic Computer is complete as it provides enough instructions in each category.

# Basic Computer Instruction Cycle

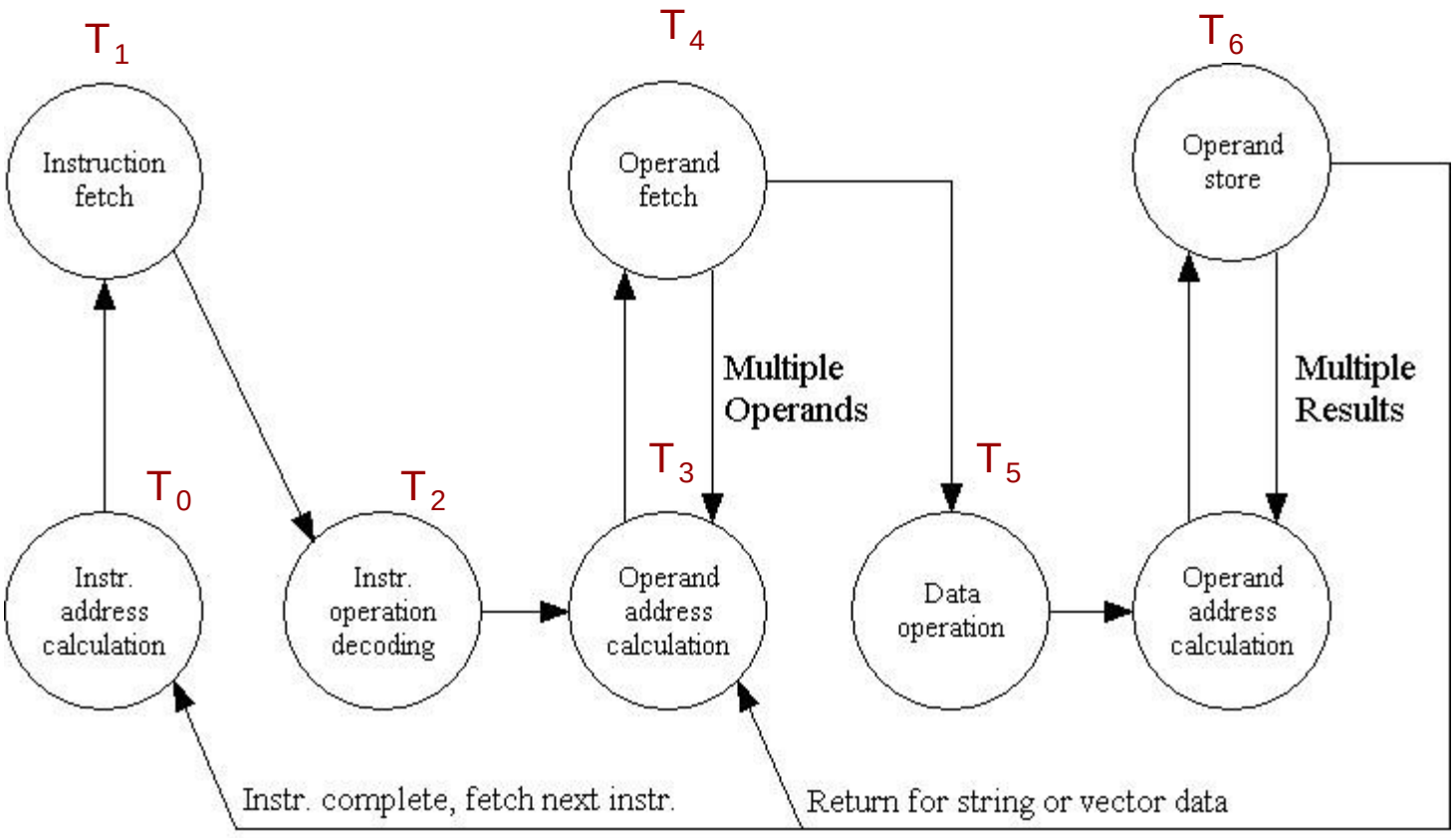
- A program residing in the memory of the computer consists of a sequence of instructions. Each instruction is executed in several steps:
  1. Fetch an instruction from memory.
  2. Decode the instruction.
  3. Read the effective address from memory if the instruction has an indirect address.
  4. Execute the instruction.
- Upon the completion of step 4, the control goes back to step 1 to start over the cycle for the next instruction.
- The process is repeated indefinitely unless a HALT instruction is encountered.

# Basic Instruction Cycle

- Computer performs the instruction cycle forever! (or at least until it is turned off, faces an error or is instructed to STOP)



# Detailed Instruction Cycle



# Detailed Instruction Cycle

- T0 - Instruction address calculation: Determines the address of the next instruction to be executed (PC)
- T1 - Instruction fetch Reads the instruction from memory location into the processor - IR
- T2 - Instruction operation decoding Analyzes the instruction to determine the type of operation to be performed and the operand(s) to be used
- T3 - Operand address calculation Determines the operand address (if needed) - AR

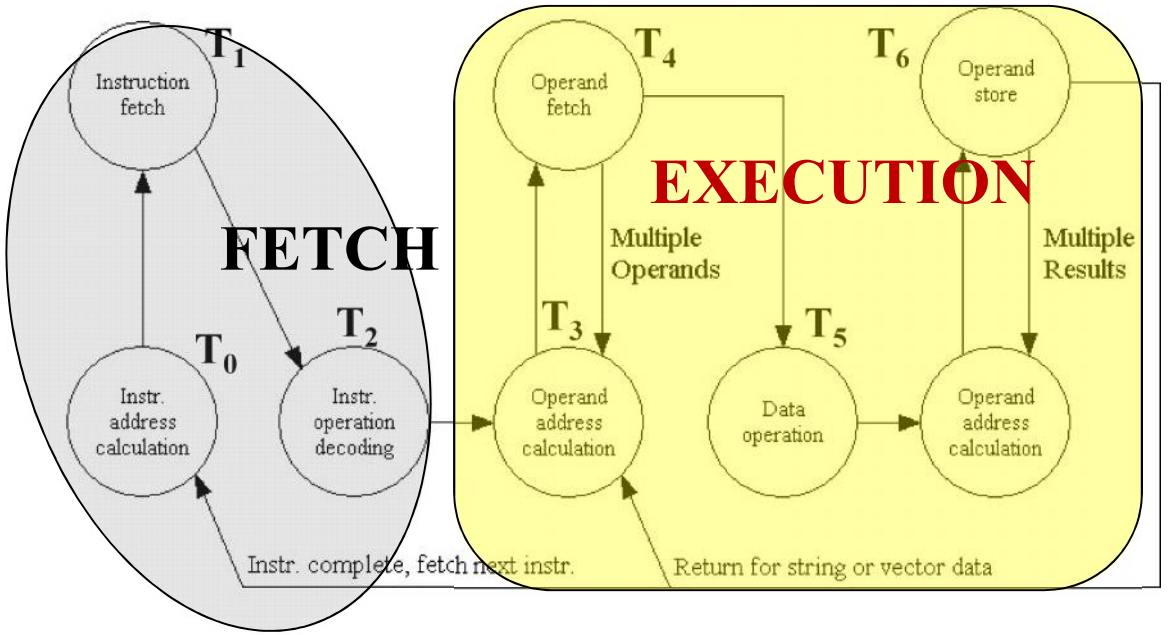
# Detailed Instruction Cycle

- T4 - Instruction execution Operand fetch: Fetches the operand from memory or reads it from I/O -> AC
- T5 – Instruction execution Data operation: Performs the operation indicated in the instruction (ALU)
- T6 - Instruction execution Operand store: Write the results into memory or out to I/O

# High Level Division of Instruction Cycle

- We first divide the instruction cycle into 2 major phases: ASM Level1 (High level of detail)

1. Fetch Phase: T0-T2
2. Execution Phase: T3-T6

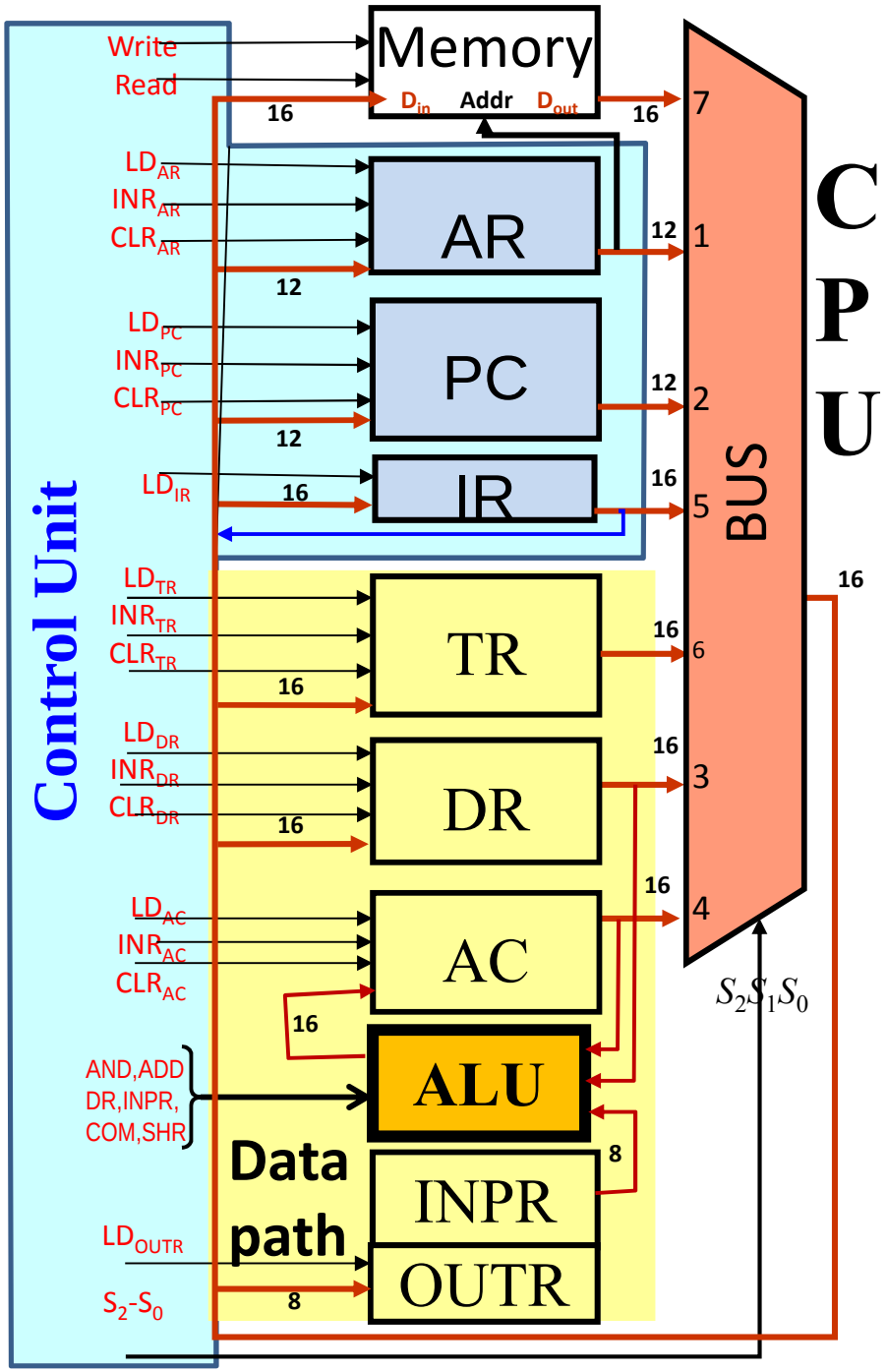


# Fetch Cycle – @ ASM Level1

- Fetch instruction from memory location specified by PC to Instruction Register (IR)
  1. $T_0$: AR ← [PC] - Transfer contents of PC to Memory Address Register
  2. $T_1$: IR <- [M(AR)] - Fetch instruction = contents of addressed memory location [M(AR)] is transferred to the IR
  3. $T_1$: PC<-PC+1 - PC incremented to point to the next instruction
  4. $T_2$ Instruction decoded by CU – now that instruction is in IR

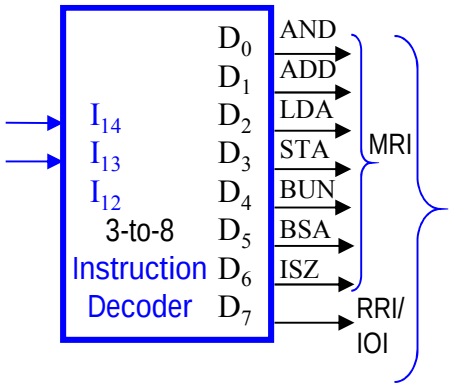
Fetch Cycle is concerned with the following registers (highlighted in light blue):

1. AR
2. PC
3. IR



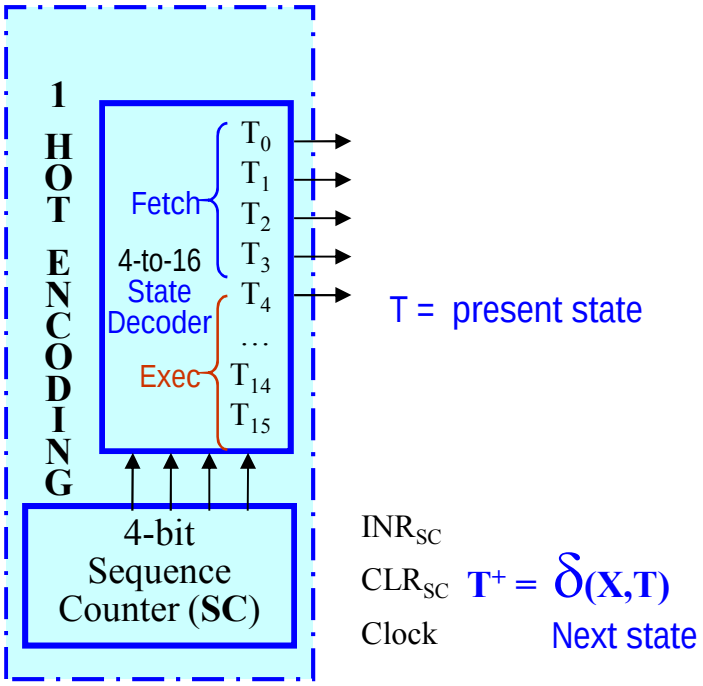
# How Instructions are Decoded?

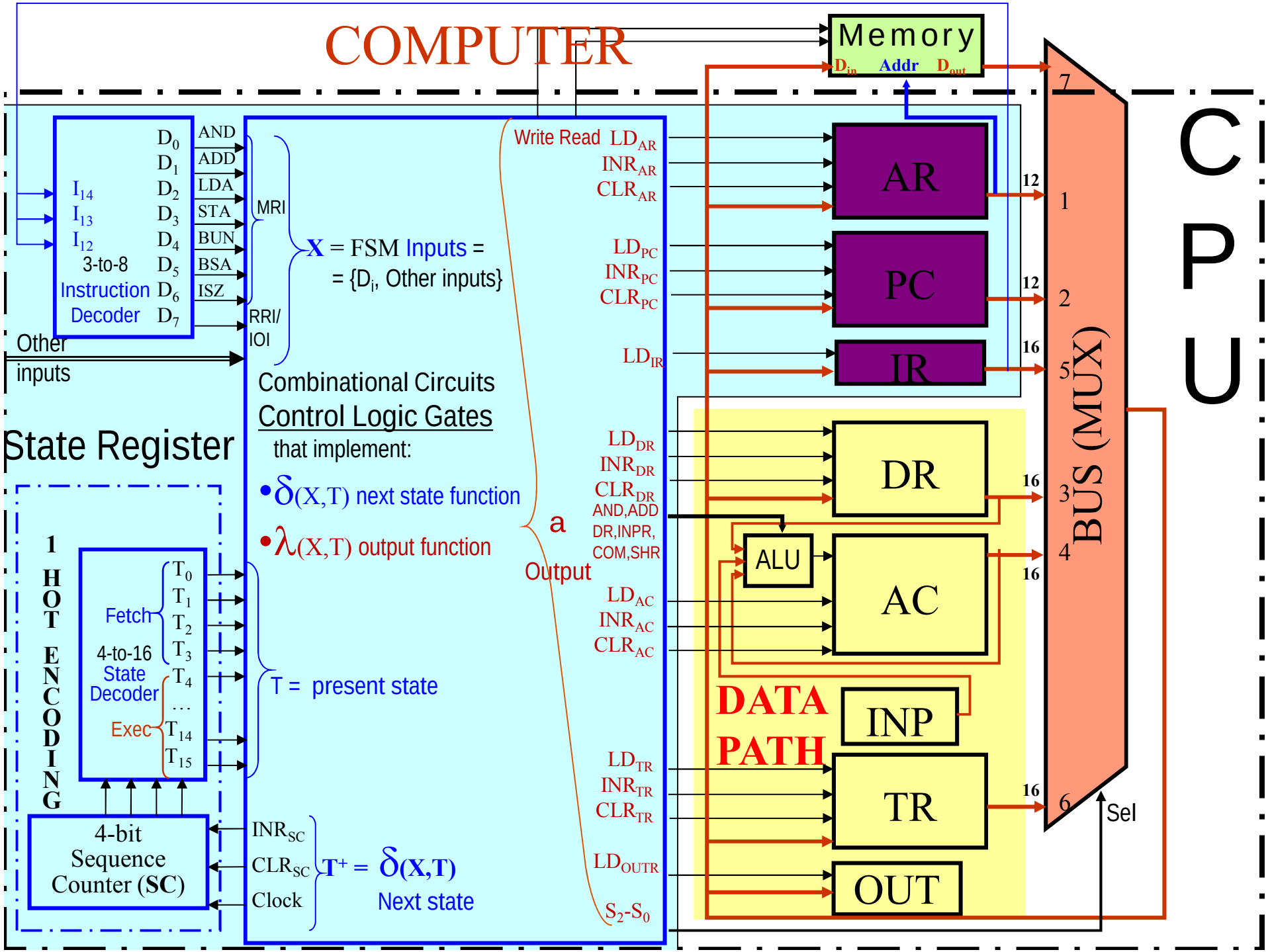
- Instructions are decoded using the Instruction Decoder
  - 3x8 decoder
  - 3bit input: bits [14:12] of the instruction register (IR)
  - 8 bit output: one-hot instruction



# How is Timing Determined?

- The Control Unit utilizes a 4-bit (modulo 16) counter and 4-to-16 decoder the determine Timing events T0-T15
  - First few timing events are used by Fetch cycle
  - Last few are used by Exec cycle



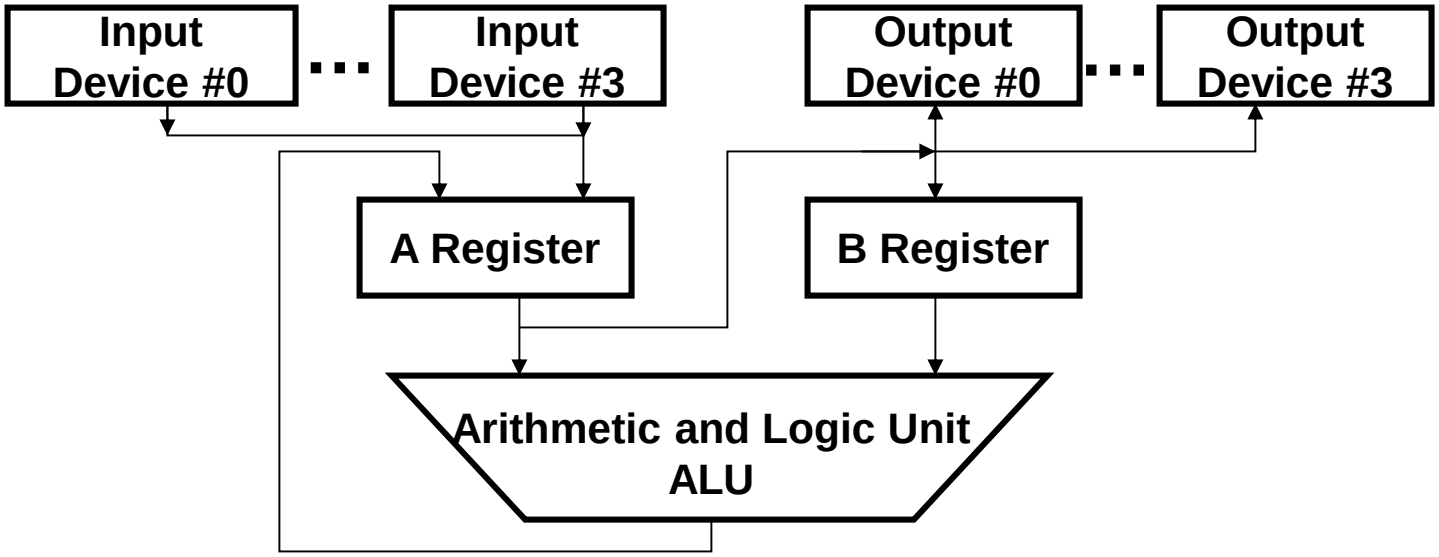

COMPUTER
Memory
$D_{in}$ Addr $D_{out}$
C P U
AR
PC
IR
DR
AC
INP
TR
OUT
ALU
DATA PATH
BUS (MUX)
7
1
2
5
3
4
6
Sel
12
12
16
16
16
16
$D_0$ AND
$D_1$ ADD
$D_2$ LDA
$D_3$ STA
$D_4$ BUN
$D_5$ BSA
$D_6$ ISZ
$D_7$ RRI/ IOI
MRI
$I_{14}$
$I_{13}$
$I_{12}$
3-to-8 Instruction Decoder
Other inputs
X = FSM Inputs = = {$D_i$, Other inputs}
Combinational Circuits
Control Logic Gates
that implement:
• $\delta(X,T)$ next state function
• $\lambda(X,T)$ output function
Write Read $LD_{AR}$ $INR_{AR}$ $CLR_{AR}$
$LD_{PC}$ $INR_{PC}$ $CLR_{PC}$
$LD_{IR}$
$LD_{DR}$ $INR_{DR}$ $CLR_{DR}$ AND,ADD DR,INPR, COM,SHR
$LD_{AC}$ $INR_{AC}$ $CLR_{AC}$
$LD_{TR}$ $INR_{TR}$ $CLR_{TR}$
$LD_{OUTR}$
$S_2$-$S_0$
a
Output
State Register
1 HOT ENCODING
$T_0$ $T_1$ $T_2$ $T_3$ Fetch
$T_4$ … $T_{14}$ $T_{15}$ Exec
4-to-16 State Decoder
T = present state
4-bit Sequence Counter (SC)
$INR_{SC}$
$CLR_{SC}$
Clock
$T^+ = \delta(X,T)$
Next state

# Pico Controller

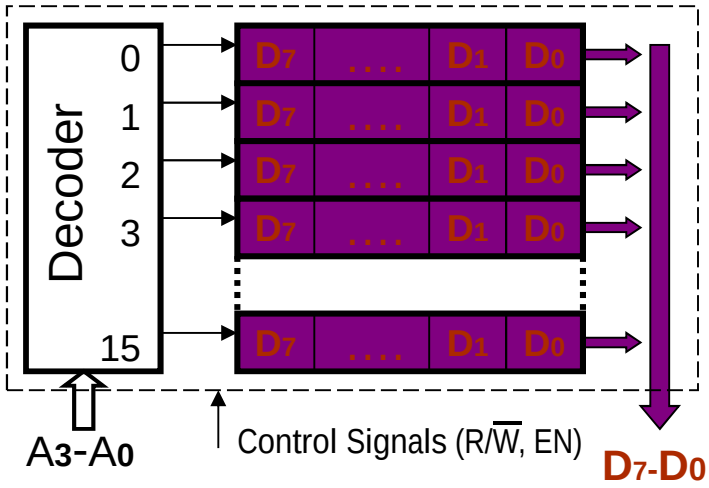
- Simple design of a processor
- Executes a few simple instructions
- Processor is basically a collection of digital logic components
  - Arithmetic Unit: support only ADD/SUB
  - 2 Accumulator Registers A & B
  - 8 bit data bus
  - Up to 4 input devices
  - Up to 4 output devices

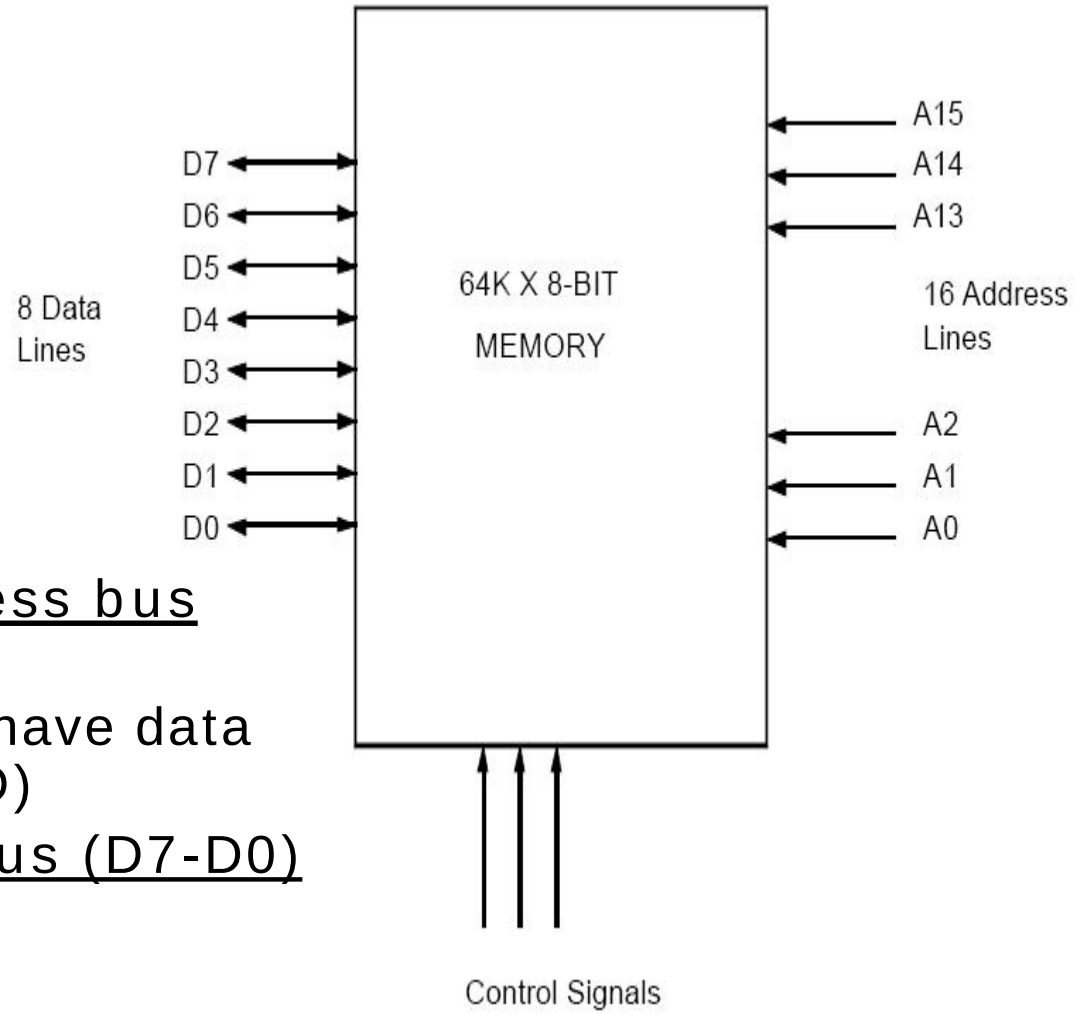
# Pico Controller Instruction Set

| Instruction | Operands | RTL | Instruction Code |
|---|---|---|---|
| ADD | N/A | A ← A + B | 001XXXXX |
| SUB | N/A | A ← A - B | 011XXXXX |
| IN | Device#, Dest Register | dd ← ii | 101Xiidd |
| OUT | Src Register, Device# | oo ← ss | 111Xssoo |
| MOV | Src & Dest Registers | dd ← ss | 010Xssdd |



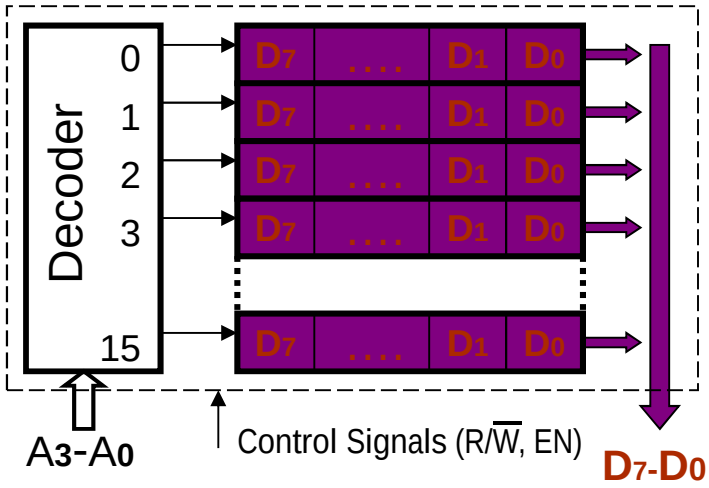
# Computer Memory

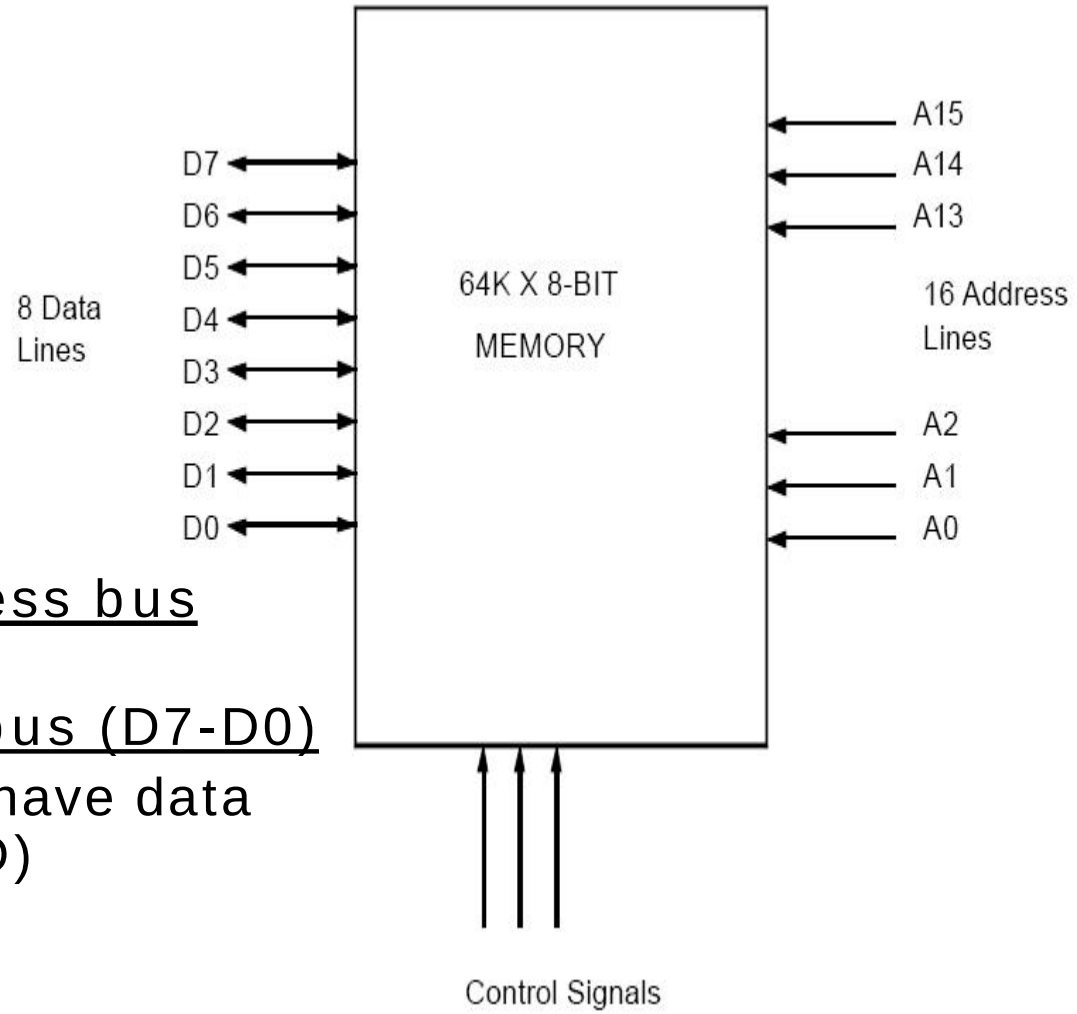



- Read operation
  - Apply address to address bus (A15-A0)
  - Apply control signal to have data appear on data bus (RD)
  - Read data from data bus (D7-D0)

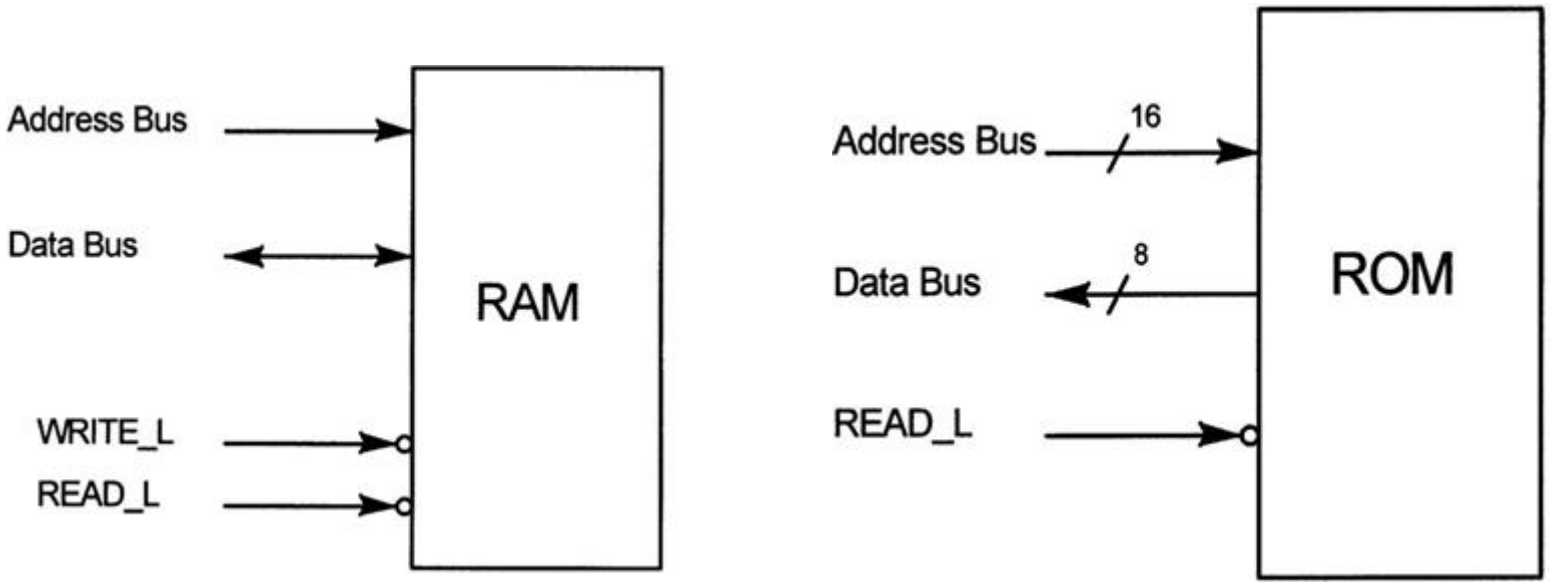
# Computer Memory




- Write operation
  - Apply address to address bus (A15-A0)
  - Apply data from data bus (D7-D0)
  - Apply control signal to have data appear on data bus (RD)

# Computer Memory – RAM & ROM




READ_L is active low read enable
WRITE_L is active low write enable

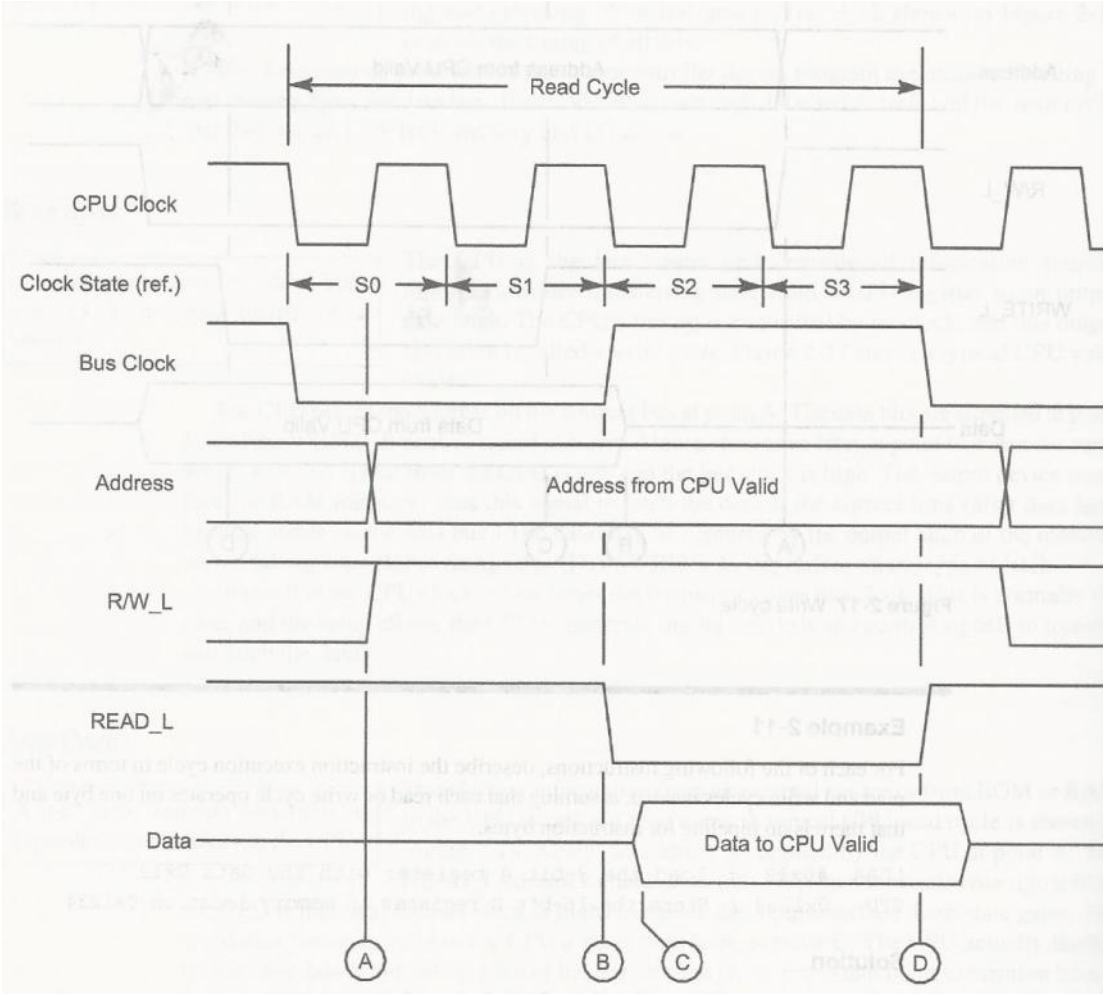
# Memory Read Cycle

*A. CPU put read address on the address bus and initiate R/W OP*

*B. At next pos-edge of bus clock, CPU set READ_L control bit*

*C. Data become available on the data bus shortly after*

*D. On neg-edge pf bus cock CPU deassert READ_L signal, RW_L is also deasserted on next pos-edge of CPU clock*



# Memory Write Cycle

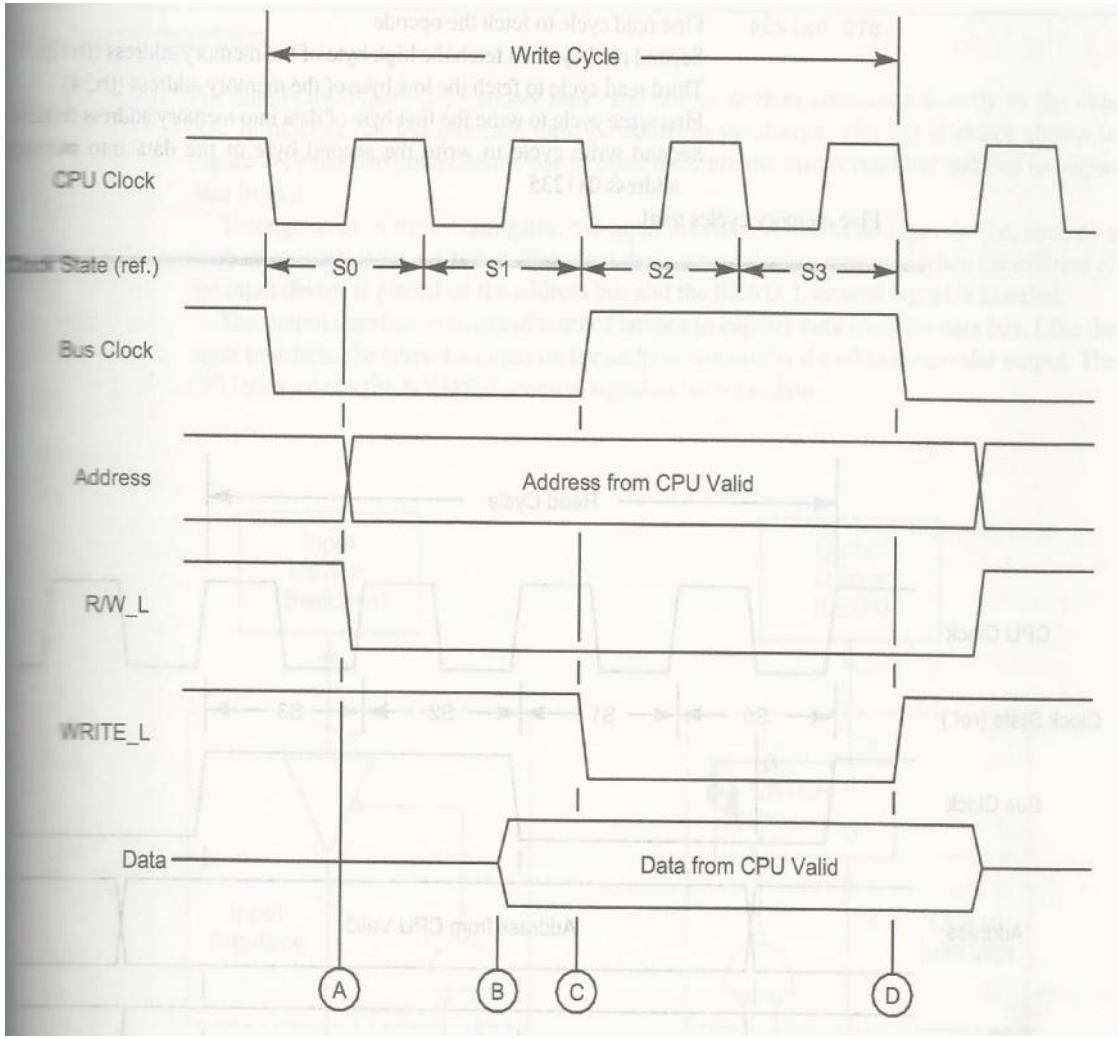
*A. CPU put read address on the address bus and initiate R/W OP*

*B. At next pos-edge of CPU clock, CPU put data on the data bus*

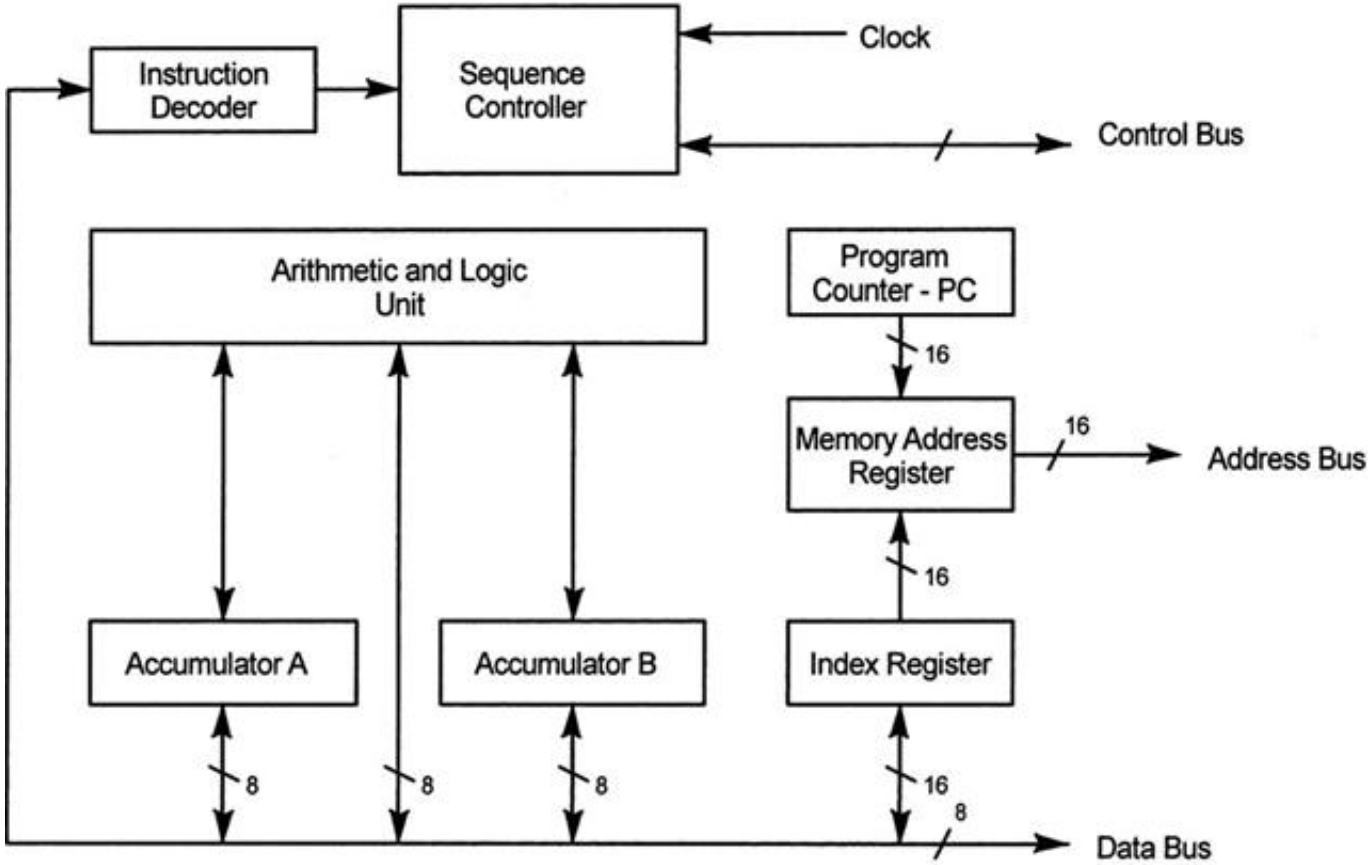
*C. On the nect pos-edge of bus clock, CPU asserts WRITE_L control signal*

*D. Data is written shortly after, and on neg-edhe of bus clock CPU de-assert WRITE_L signal*

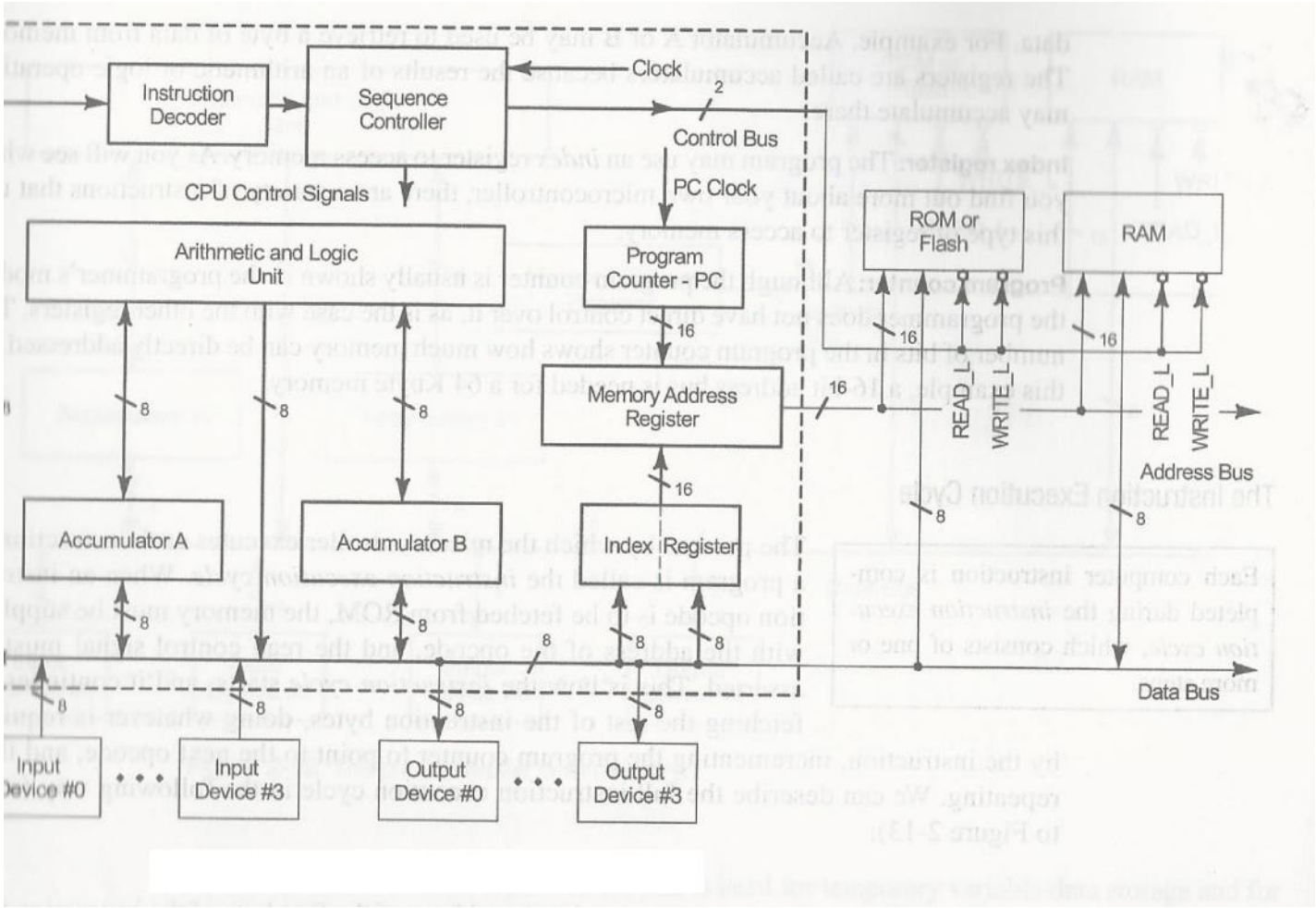
*E. On next pos-edge of CPU clock CPU de-assert RW_L signal*



# Pico Controller CPU



# Connecting RAM & ROM to the CPU



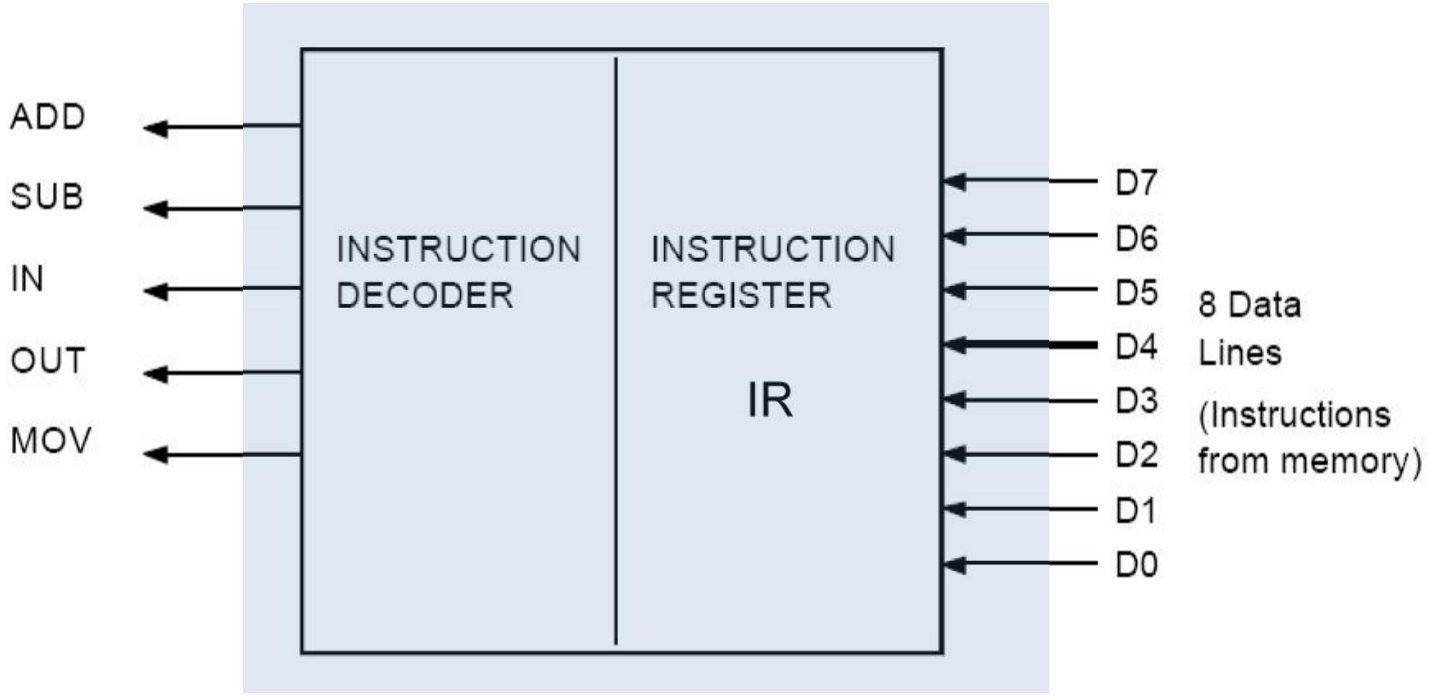
# Program in Computer Memory

| | Operation Field | | Comment Field | Addr | **Contents** |
|---|---|---|---|---|---|
| 1 | IN | 1 | (A) ← (Switches) | 0: | 1010 0001 |
| 2 | MOV | | (B) ← (A) | 1: | 0100 0000 |
| 3 | IN | 1 | (A) ← (Switches) | 2: | 1010 0001 |
| 4 | ADD | | (A) ← (A) + (B) | 3: | 0010 0000 |
| 5 | OUT | 2 | (LED) ← (A) | 4: | 1110 0010 |

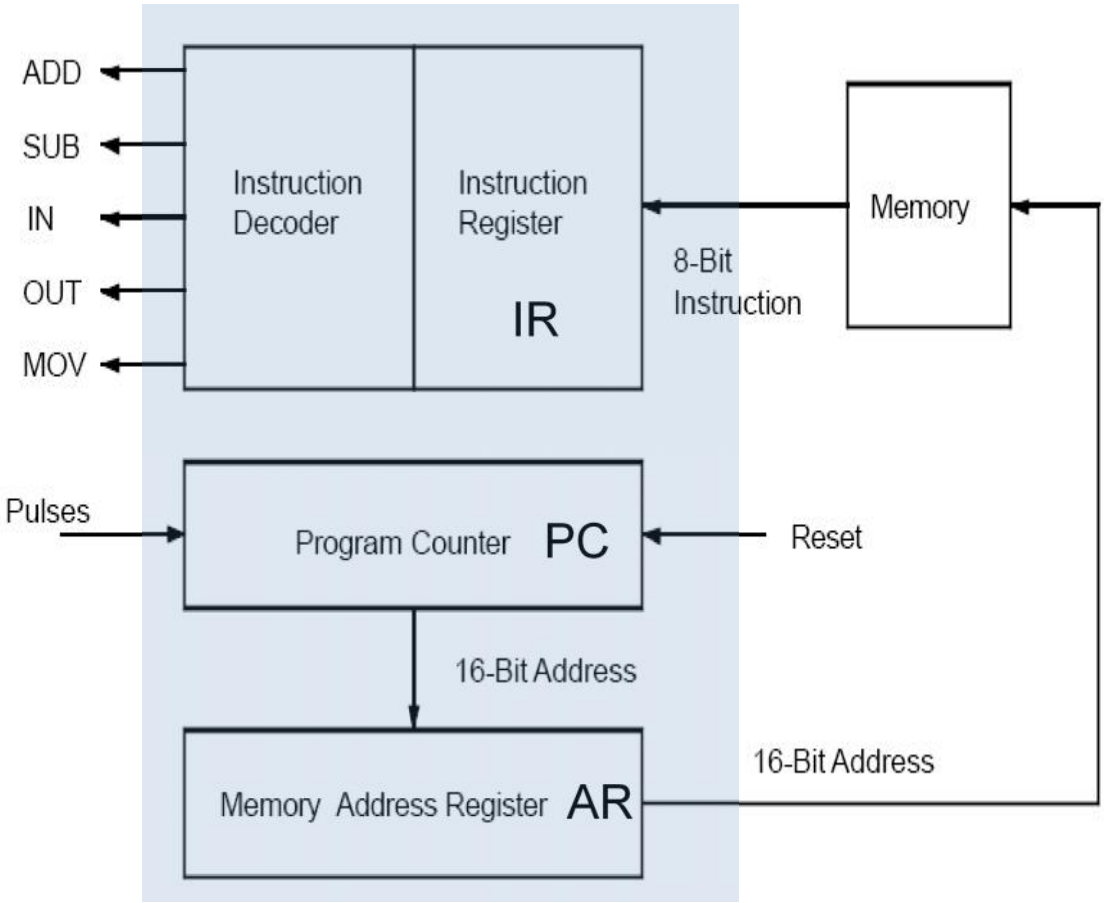
# Instruction Register and Decoder

- Instruction register
  - Used to hold the computer instruction
- Instruction decoder
  - Logic circuit that decodes instruction
  - Asserts logic signals for each operation – example IN used as clock signal to latch data into the A register
  - Other logic signals not shown – decode OUT and IN operands to select devices

# Instruction Register and Decoder



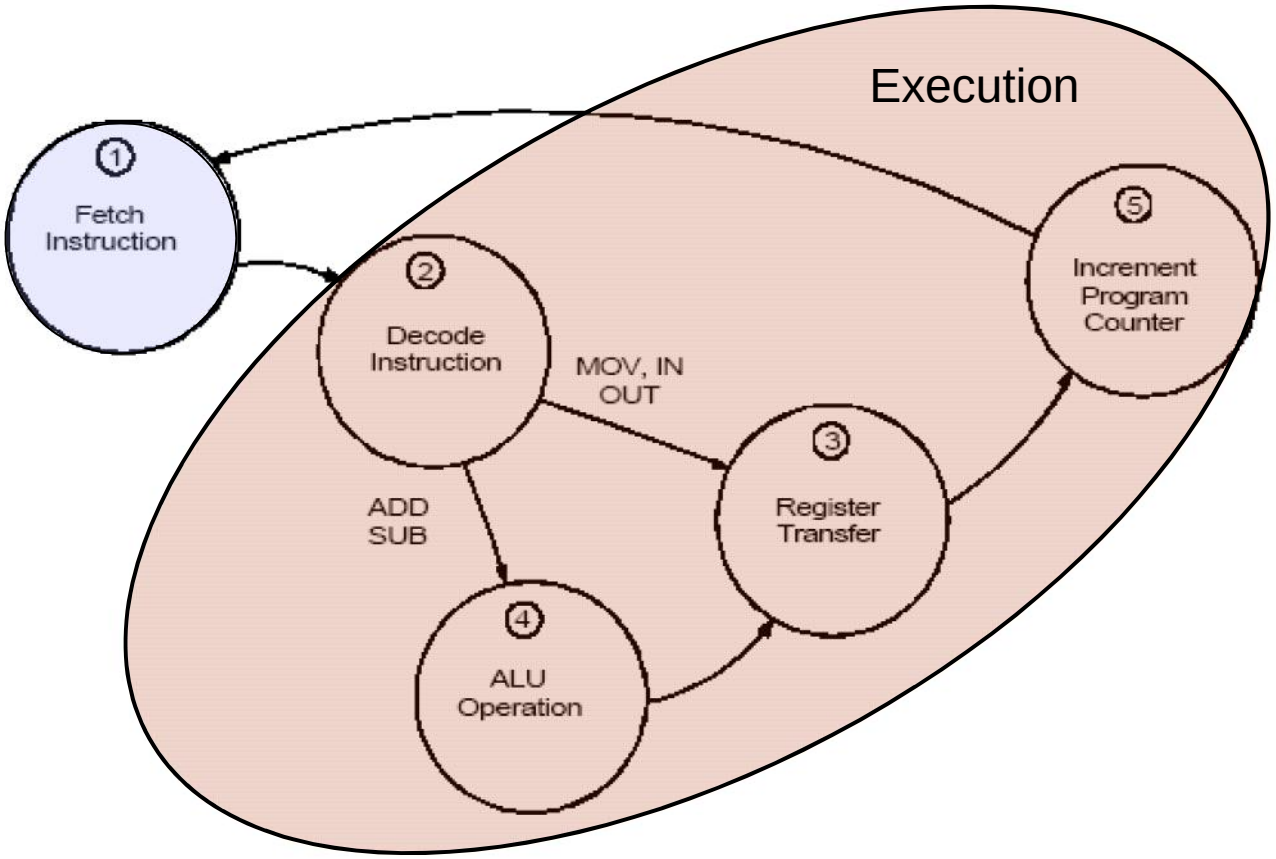
# Program Counter & Address Register



# IN Instruction Execution Cycle

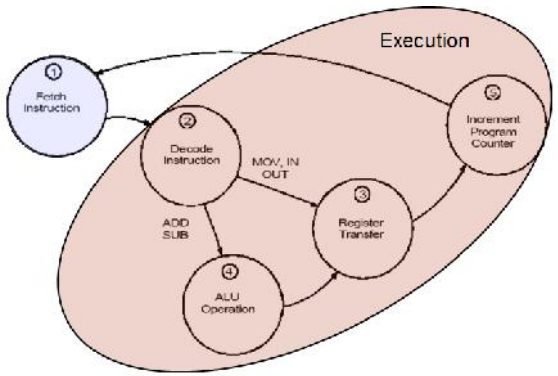
- Need to start thinking about time it takes to execute instructions by considering the steps to execute an instruction
- Fetch instruction
  - Transfer contents of PC to Memory Address Register
  - Contents of addressed memory location transferred to Instruction Register (instruction decoded)
  - Instruction decoder asserts IN signal line
- Execution
  - Data in switches are transferred to A register
  - PC incremented to 0001
- Process repeated for next instruction

# CPU State Machine



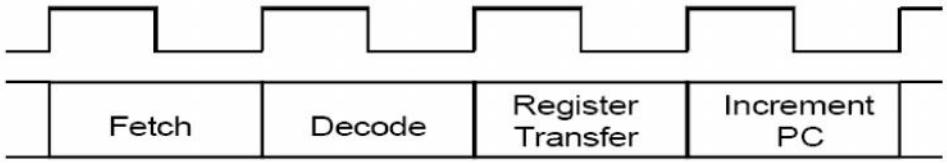
# CPU State Machine

- Break down instruction into small elements - states
  - Each element-state takes an amount of time
  - Allows time for event to occur
  - Event: Propagation of signals
- Sequence Controller:
  - Implements state machine
  - With Instruction decoder, generates control signals at appropriate time

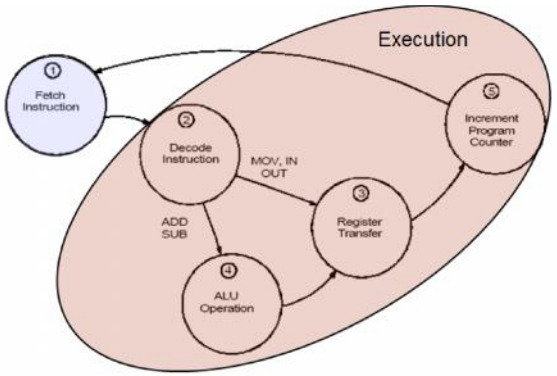


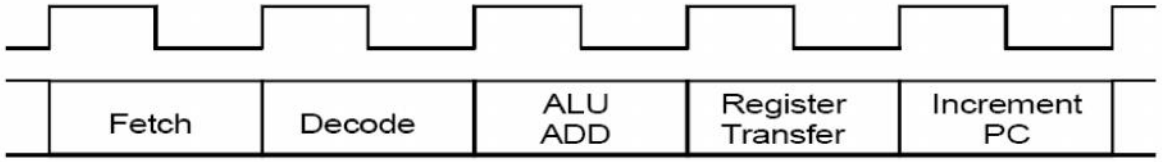
# System Timing

- Sequential state machine operated by clock
- Consider 1 MHz clock –
- Pulse every 1 μsec for each state




(a) Timing for the IN instruction.




(b) Timing for the ADD instruction.

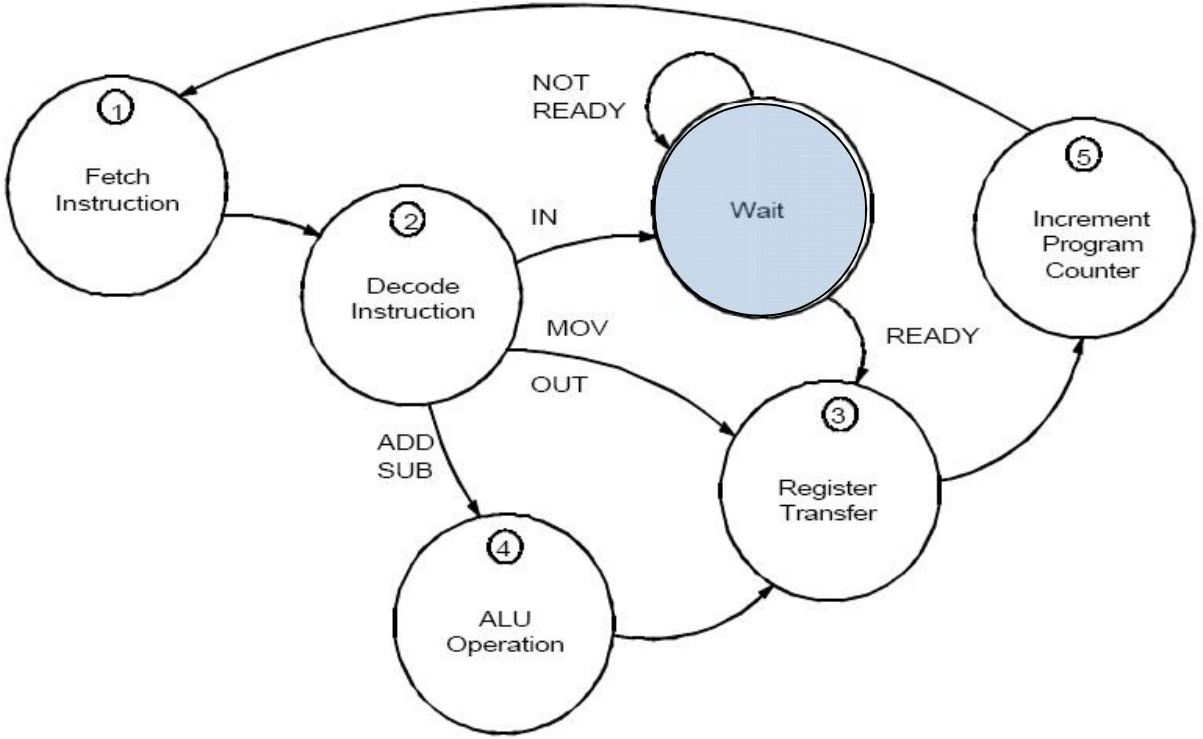
# Program Execution Time

- Using 1 MHz clock, following table shows time to execute program

| Instruction | Number of states |
|---|---|
| IN 1 | 4 |
| MOV | 4 |
| IN 1 | 4 |
| ADD | 5 |
| OUT 2 | 4 |
| Total states | 21 = 21 μsec |

# I/O Synchronization and Wait States

- Wait state is used to synchronize with external event
  - Deals with slower devices (e.g. switches)

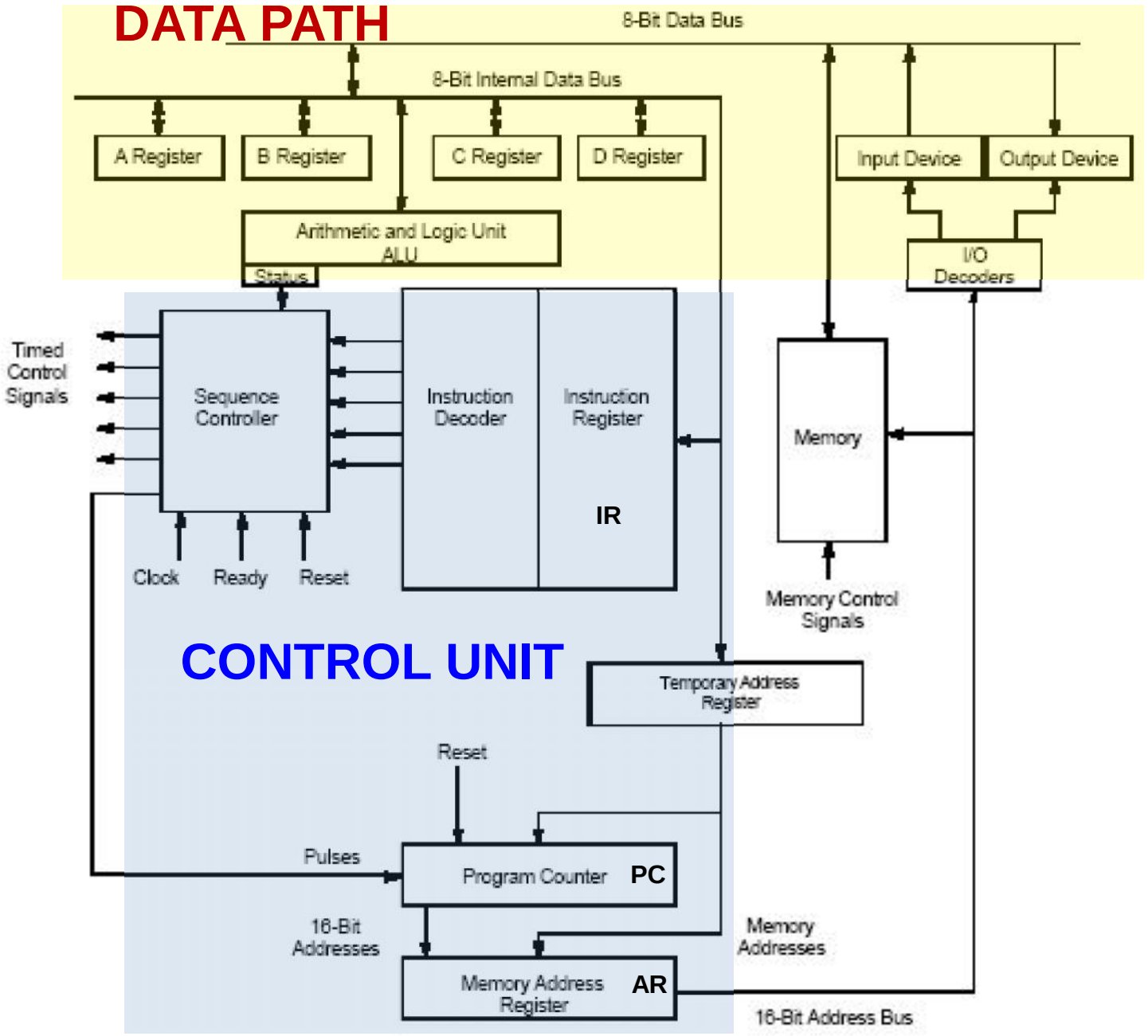



*Note: We can also add wait state for OUT instruction*

# What is missing

- Memory Reference instructions
  - Using different addressing modes
- Program control instructions
- Interrupts
- What else?!
- Will cover these later..

# Final Desig


DATA PATH
8-Bit Data Bus
8-Bit Internal Data Bus
A Register
B Register
C Register
D Register
Input Device
Output Device
Arithmetic and Logic Unit
ALU
Status
I/O
Decoders
Timed
Control
Signals
Sequence
Controller
Instruction
Decoder
Instruction
Register
IR
Memory
Clock
Ready
Reset
Memory Control
Signals
CONTROL UNIT
Temporary Address
Register
Reset
Pulses
Program Counter
PC
16-Bit
Addresses
Memory
Addresses
Memory Address
Register
AR
16-Bit Address Bus

# References


- Fredrick M. Cady, Microcontrollers and Microcomputers: Principles of Software and Hardware Engineering – Chapter 1 & 2