CS523
Top Down + Bottom UP
Understanding a Processor and Designing a Processor

Dr. A. Sahu
CSE IIT Guwahati

Books : Text

- Patterson, D.A., and Hennessy, J.L. , “**Computer Architecture : A Quantitative Approach** ”, Morgan Kaufmann Publishers, 5th Edition, Inc.2011

- Dezso Sima, Peter Kacsuk, Terence Fountain, " **Advanced Computer Architectures : A Design Space Approach**", Pearson Education India, 1997

- Michael J Flynn, "**Computer Architecture: Pipelined and Parallel Processor Design** ", Narosa Publishing India, 2003

- Some Recent Papers

Books: References

- Patterson, D.A., and Hennessy, J.L. , "**Computer Organization and Design: The Hardware/Software Interface**", Morgan Kaufmann Publishers, 4th Edition, Inc.2005, Ebook 3rd Ed
- Kai Hwang, " **Advanced Computer Architecture: Parallelism, Scalability, Programmability**", McGraw-Hill, first edition, 1992.
- Ramachandran Vaidtyanathan and J L Trahan, " **Dynamic Reconfiguration: Architectures and Algorithms** ", Kluwer Academic Publisher, New York, 2003
- David Kirk and Wen-mei Hwu " **Programming Massively Parallel Processors: A Hands-on Approach**", Morgan Kaufmann Publishers, 2010
- P Pacheco " **An Introduction to Parallel Programming**", Morgan Kaufmann Publishers, 2011
- David Culler, J.P. Singh and Anoop Gupta, "**Parallel Computer Architecture: A Hardware/Software Approach**", Morgan Kaufmann, first edition, 1998.
- Harvey G Cragon, " **Memory Systems and Pipelined Processors**", Narosa Book Distributors, India, 1998
- **Introduction to Parallel Programming**", Morgan Kaufmann Publishers, 2011

Data Parallel Architectures

- SIMD Processors
  – Multiple processing elements driven by a single instruction stream
- Vector Processors
  – Uni-processors with vector instructions
- Associative Processors
  – SIMD like processors with associative memory
- Systolic Arrays
  – Application specific VLSI structures

GPU Architecture

- Mix of Data Parallel and Function Parallel
  - Only limited functions can be run (Kepler Architecture)
  - Other wise it is data parallel
- How to execute a Program
  - Load data and function to GPU
  - Triggers GPU
  - Get back result
- GIGA thread scheduler
- Nvidia GTX 3072 cuda core, 8 thread/core
- Many shared data/global object/PC data

Systolic Arrays [H.T. Kung 1978]

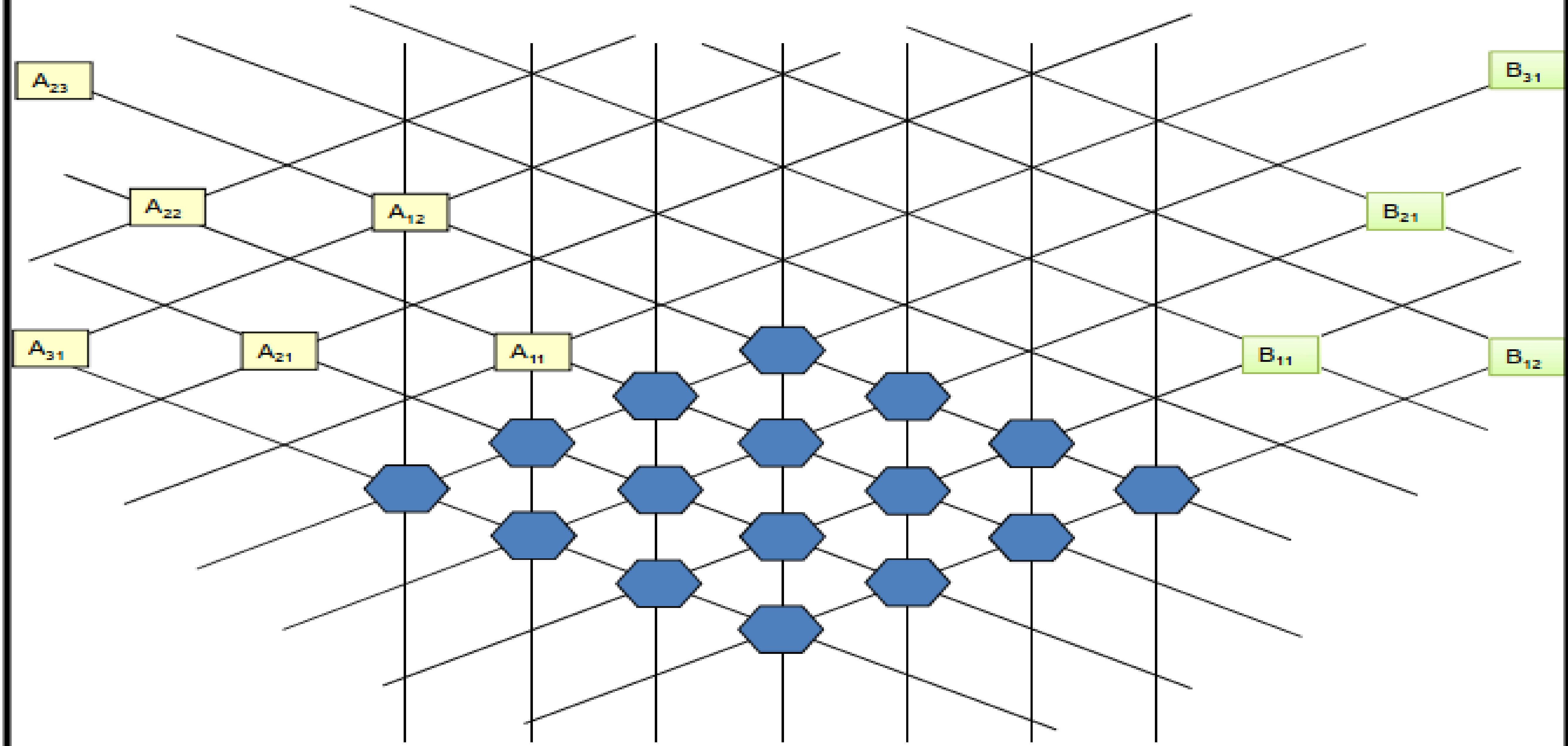
Simplicity, Regularity, Concurrency, Communication

Example :

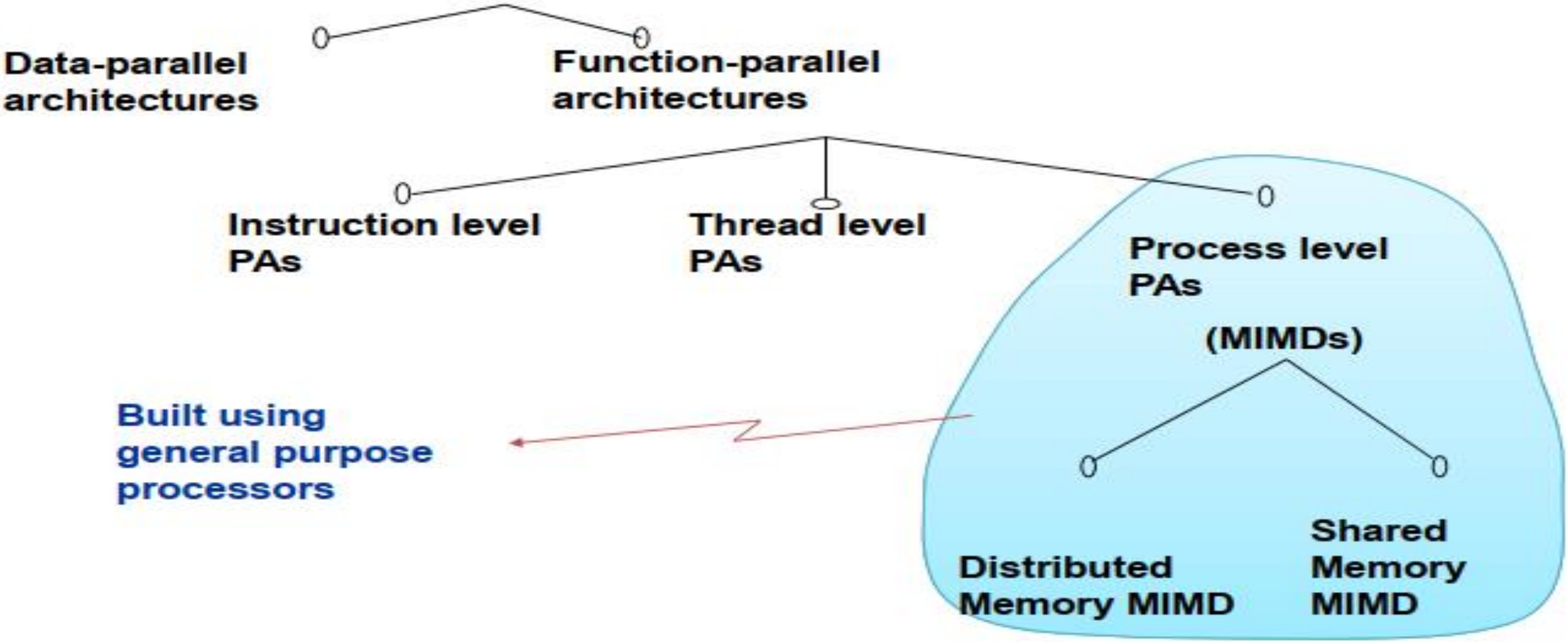
Band matrix multiplication

$$[C] = \begin{bmatrix} A_{11} & A_{12} & 0 & 0 & 0 & 0 \\ A_{21} & A_{22} & A_{23} & 0 & 0 & 0 \\ A_{31} & A_{32} & A_{33} & A_{34} & 0 & 0 \\ 0 & A_{42} & A_{43} & A_{44} & A_{45} & 0 \\ 0 & 0 & A_{53} & A_{54} & A_{55} & A_{56} \\ 0 & 0 & 0 & A_{64} & A_{65} & A_{66} \end{bmatrix} \bullet \begin{bmatrix} B_{11} & B_{12} & 0 & 0 & 0 & 0 \\ B_{21} & B_{22} & B_{23} & 0 & 0 & 0 \\ B_{31} & B_{32} & B_{33} & B_{34} & 0 & 0 \\ 0 & B_{42} & B_{43} & B_{44} & B_{45} & 0 \\ 0 & 0 & B_{53} & B_{54} & B_{55} & B_{56} \\ 0 & 0 & 0 & B_{64} & B_{65} & B_{66} \end{bmatrix}$$

T=0



Why Process level Parallel Architectures?



MIMD Architectures

Design Space

- Extent of address space sharing
- Location of memory modules
- Uniformity of memory access
  - UMA, NUMA

Issues from user's perspective

- Specification / Program design
  - explicit parallelism or
  - implicit parallelism + parallelizing compiler
- Partitioning / mapping to processors
- Scheduling / mapping to time instants
  - static or dynamic
- Communication and Synchronization

Parallel programming models

CUDA
Pthread
MPI
Cilk

Concurrent control flow

Functional or logic program

Vector/array operations

Concurrent tasks/processes/threads/objects

With shared variables or message passing

Relationship between programming model and architecture ?

Issues from architect's perspective

- Coherence problem in shared memory with caches
  - Software Coherence : Synchronization and Lock
  - Lock: One person should access
  - Critical Section, Amdhal's Law, Modified Law
- Efficient interconnection networks

Cache Coherence Problem

Multiple copies of data may exist
$\Rightarrow$ Problem of cache coherence

Options for coherence protocols

- What action is taken?
  - Invalidate or Update
  - Lazy protocol
- Which processors/caches communicate?
  - Snoopy (broadcast) or directory based
- Status of each block?
- **Memory Consistency Problem: Pthread Level**

Interconnection Networks

- Architectural Variations:
  - Topology
  - Direct or Indirect (through switches)
  - Static (fixed connections) or Dynamic (connections established as required)
  - Routing type store and forward/worm hole)
- Efficiency:
  - Delay
  - Bandwidth
  - Cost

Processor Design : Basic

- Microprocessor Vs Processor
  - Fetch, Decode, Register Access, Execute, Write Back
  - Single Cycle Design : All state in one cycle
  - Multi Cycle Design: Each state in one cycle Multiple T State, 8085 Microprocessor
- 8085 Microprocessor
- RISC Processor
- Simple MIPS Processor with 9 Instructions
- One Instruction Set Computer

To be discussed

Understanding a given Processor
(Example 8085)

Designing a Processor
(Example Tiny MIPS (9 Instructions))

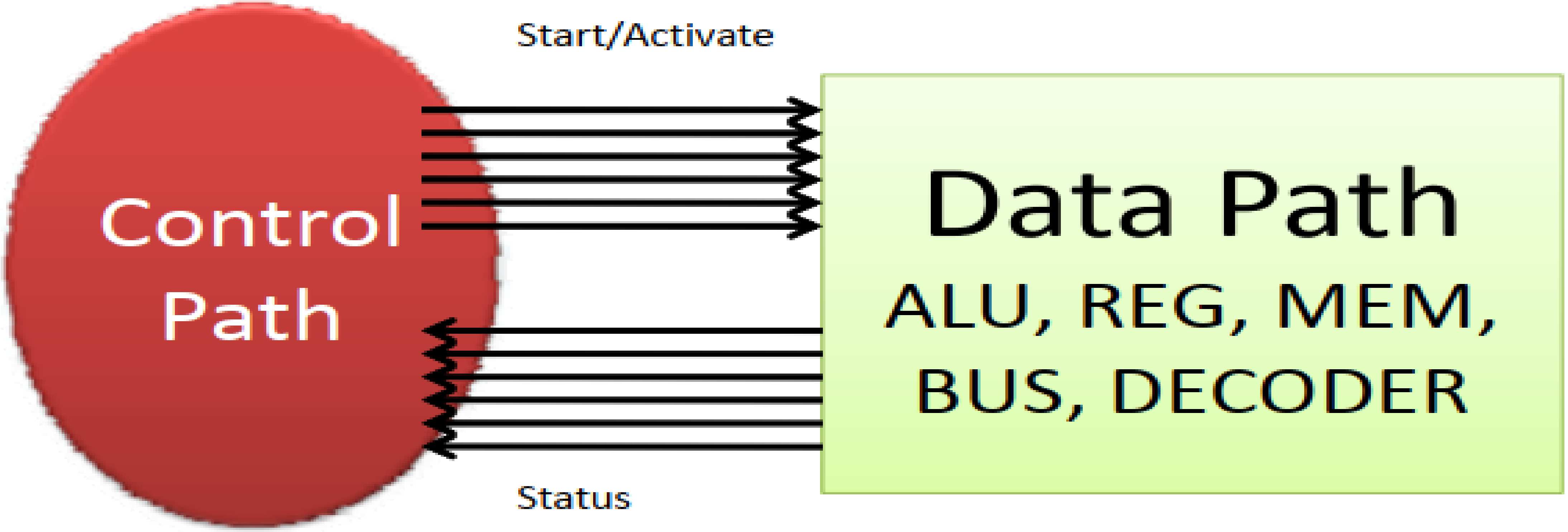
Understanding a given Processor

(Example 8085)

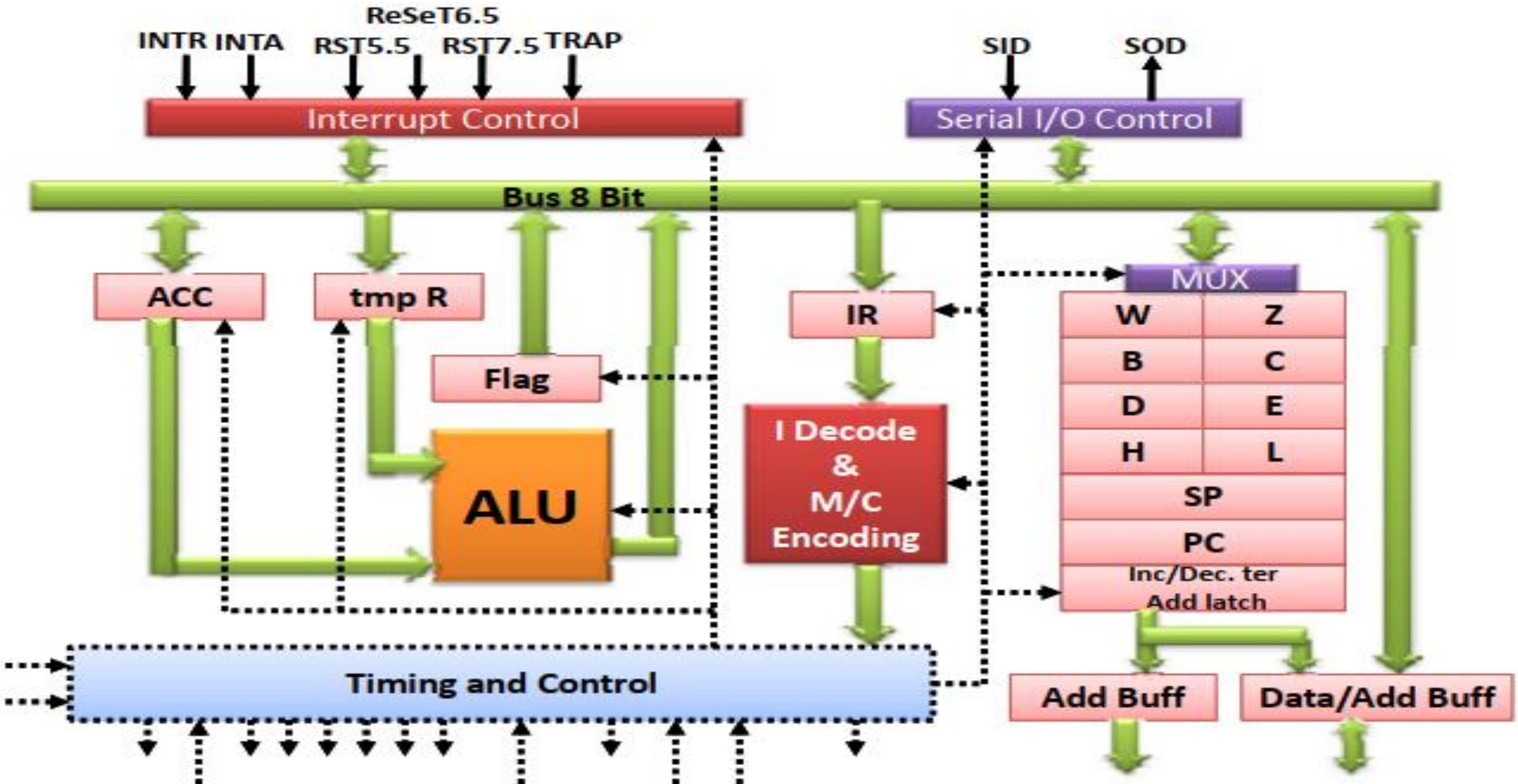
8085 Microprocessor

- 8 Bit CPU
- 3-6Mhz
- Simpler design: Single Cycle CPU
- ISA = Pre x86 design (Semi CISC)
- 40 Pin Dual line Package
- 16 bit address
- 6 registers: B, C, D, E, H,L
- Accumulator 8 bit

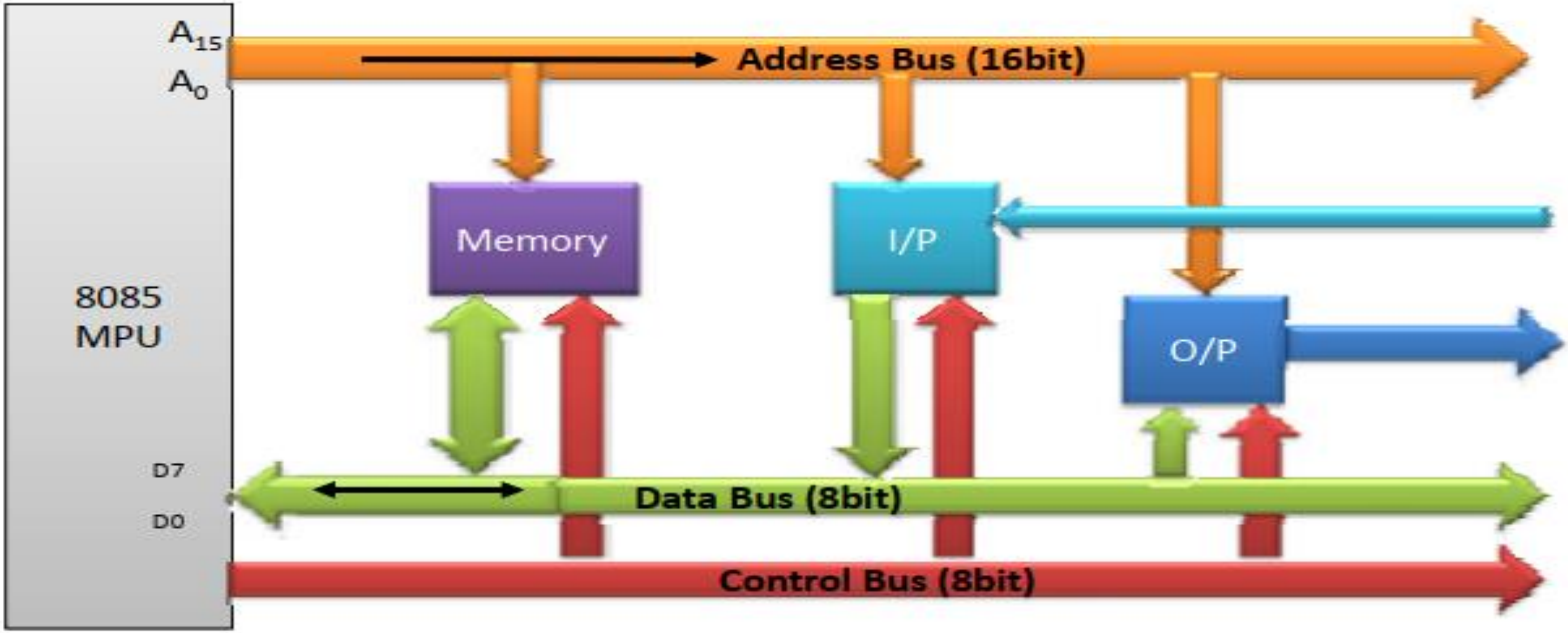
Processor


Start/Activate
Control Path
Data Path
ALU, REG, MEM,
BUS, DECODER
Status

8085 Microprocessor Architecture



The 8085 Bus Structure



8085 Bus Structure

- Address Bus : Consists of 16 address lines: $A_0 - A_{15}$
  - Address locations: 0000 (hex) – FFFF (hex)
  - Can access 64K (= 2^{16}) bytes of memory, each byte has 8 bits
  - Can access 64K × 8 bits of memory
  - Use memory to map I/O, Same instructions to use for accessing I/O devices and memory
- Data Bus : Consists of 8 data lines: $D_0 - D_7$
  - Operates in bidirectional mode
  - The data bits are sent from the MPU to I/O & vice versa
  - Data range: 00 (hex) – FF (hex)
- Control Bus:
  - Consists of various lines carrying the control signals such as read / write enable, flag bits

8085 Registers

- Registers:
  - Six general purpose 8-bit registers: B, C, D, E, H,L
  - Combined as register pairs to perform 16-bit operations: BC, DE, HL
  - Registers are programmable (load, move, etc.)
- Stack Pointer (SP)
- Accumulator & Flag Register
  - (Zero, Sign, Carry, Parity, AuxCarry)
- Program Counter (PC)
  - Contains the memory address (16 bits) of the instruction that will be executed in the next step.

B	C
D	E
H	L
SP	
PC	

How instruction executed

- All instructions (of a program) are stored in memory.
- To run a program, the individual instructions must be read from the memory in sequence, and executed.
  - Program counter puts the 16-bit memory address of the instruction on the address bus
  - Control unit sends the Memory Read Enable signal to access the memory
  - The 8-bit instruction stored in memory is placed on the data bus and transferred to the instruction decoder
  - Instruction is decoded and executed

Instruction Set of 8085

- Arithmetic Operations
  - add, sub, inr/dcr
- Logical operation
  - and, or, xor, rotate, compare, complement
- Branch operation
  - Jump, call, return
- Data transfer/Copy/Memory operation/IO
  - MOV, MVI, LD, ST, OUT

Copy/Mem/IO operation

- MVI R, 8 bit *// load immediate data*
- MOV R1, R2 *// Example MOV B, A*
- MOV R M *// Copy to R from 0(HL Reg) Mem*
- MOV M R *// Copy from R to 0(HL Reg) Mem*

- LDA 16 bit *// load A from 0(16bit)*
- STA 16 bit *// Store A to 0(16bit)*
- LDAX Rp *// load A from 0(Rp), Rp=RegPair*
- STAX Rp *// Store A to 0(Rp)*
- LXI Rp 16bit *// load immediate to Rp*

- IN 8bit *// Accept data to A from port 0(8bit)*
- OUT 8 bit *// Send data of A to port 0(8bit)*

Arithmetic Operation

- ADD R // *Add A = A + B.reg*
- ADI 8bit // *Add A= A + 8bit*
- ADD M // *Add A=A + O(HL)*

- SUB R // *Sub A = A -B.reg*
- SUI 8bit // *Sub A= A - 8bit*
- SUB M // *Sub A=A - O(HL)*

- INR R // *R = R+1*
- INR M // *O(HL)=O(HL)+1*
- DCR R // *R = R-1*
- DCR M // *O(HL)=O(HL)-1*
- INX Rp // *Rp=Rp+1*
- DCX Rp // *Rp=Rp-1*

Other Operations

- Logic operations
  - ANA R   ANI 8bit   ANA M
  - ORA, ORI, XRA, XRI
  - CMP R // compare with R with ACC
  - CPI 8bit // compare 8 bit with ACC
- Branch operations
  - JMP 16bit, CALL 16 bit
  - JZ 16bit, JNZ 16bit, JC 16bit, JNC 16 bit
  - RET
- Machine Control operations
  - HLT, NOP, POP, PUSH

Instruction to Micro-Instructions

- ADD B // ACC = ACC + B
- Things need to do to execute this
  - Fetch Instruction from Memory and put into IR
    - Put Higher address to A8-15
    - Put Lower Address to A0-A7
    - Get back data/Instruction from memory to BUS
    - Activate BUS to IR gate to store date in IR
- Addition need to be done in ALU
  - Operands should be in TempR and ACC, Result will put to BUS
- Execute
  - Activate MUX to select Register B
  - Put value to B to BUS
  - Activate Gate of Temp Reg, so that data from BUS will go to TempR
  - Do the Operation ADD
- Again result from BUS put to ACC
  - Activate Gate for ALU, so that data from ALU come to BUS
  - Activate Gate for ACC, so that data from BUS go to ACC

Designing a Processor

(Example Tiny MIPS (9 Instructions))

Next Class……