Processor Memory(Registers)

PROCESSOR MEMORY (REGISTERS)

Microprocessor Registers

- Microcomputer memory system can logically be divided into three groups:
  - Processor memory
  - Primary or main memory
  - Secondary memory

Cont.

- Processor memory refers to a set of microprocessor registers. These registers form storage locations in a CPU.
- They are used to hold data/addresses when execution of an instruction is in progress.
- Processor registers and the microprocessor are fabricated using the same technology. They have faster access time than primary memory.

Cont.

- The number of registers available for programming in a microprocessor vary with different processors.
- 8085 microprocessor registers are:

  The Accumulator(A), B,C,D,E,H,L, Flag Register, Stack Pointer(SP),Program Counter(PC).

Program Counter(PC)

- It is 16 bits long.
- The Program Counter is used to address up to 64K of memory. It is always holding the address of the next instruction to be fetched from memory.
- It enables the processor to read successive instructions that are stored in the memory.
- PC is modified with new address when there is a **jump**, **transfer of control** or **subroutine call** instructions.

Stack Pointer(SP)

- A Stack is an area in memory(RAM) that is used for temporary storage of data or return addresses.
- A SP is a register that points to the next free location in the stack.
- Each time a byte is put onto the stack, SP is decremented.
- Each time a byte is retrieved from the stack, SP is incremented.

Cont.

- The PUSH command is used to store contents in the stack and the POP command retrieves stored contents.

Programming

- A microprocessor is driven by a set of instructions to perform specific tasks.
- An instruction is a statement that becomes executable when a program is assembled
- A program can be written in
  - Machine Language,
  - Low Level Language(Assembly Language),
  - High Level Language such C, Java
- Instructions are translated by the assembles into machine language bytes, which are loaded and executed by the CPU at runtime.

cont.

- Assembly can be done manually by the programmer looking up the codes in the instructions set (hand assembly) or by the machine using the assembler.

- An instructions set contains all the instructions that a particular type of microprocessor can execute.

Instructions

- An instruction contains four basic parts;
  - Label (optional)
  - Instruction Mnemonic (required)
  - Operand(s) (usually required)
  - Comment (optional)

- The different parts of an instruction is arranged as shown bellow;

[Label:] mnemonic [operands] [; comment]

- Label is an identifier that acts as a place marker for instructions and data. A label placed just before an instruction implies the instruction's address. Similarly , a label placed just before a variable implies the variable’s address.
- There are two types of labels; Data labels and Code labels

- A data label identifies the location of a variable, providing a convenient way to reference the variable in code. The following , for example, defines a variable named count:

```
count  DWORD   100
```

It is possible to define multiple data items following a label. In the following example , *array* defines the location of the first number(1024). The other numbers will follow in memory immediately afterwards

```
array   DWORD  1024 ,   2048
        DWORD   4096,   8192
```

A label in code area of a program (where intructions are located) must end with (:) character. Code labels are used as targets of jumping and looping instructions.

- For example, the following JMP (jump) instruction transfers control to the location named **target**, creating a loop

```
target:
    mov      ax, bx
    ……
    jmp      target
```

- A code label can share the same line with an instruction, or it can be on a line by itself:

```
L1:
L2:    mov         ax,  bx
```

Instruction Mnemonic

- An instruction mnemonic is a short word that identifies an instruction. In English, a mnemonic is a device that assist memory. Similarly, assembly language instruction mnemonics such as ***mov***, ***add***, and ***sub*** provide hints about the type of operation they perform .
- Following are examples of instruction mnemonics

Mnemonic	**Description**
MOV	move (assign) one value to another

ADD	Add two values
SUB	subtract one value from another
JMP	Jump to a new location
CALL	Call a procedure

Operands

- An operand is a value that is used for input or output for an instruction. Assembly language instruction can have between zero and three operands, each of which can be a register, memory operand, integer expression, or input-output port.
- The STC instruction, for example, has no operands

```
  stc                ;set Carry flag
```

The INC has one operand;

```
 inc        eax    ;add 1 to EAX
```

- The MOV instruction has two operands:
- MOV count, ebx ;move ebx to count

- There is a natural ordering of operands. When instructions have multiple operands, the first one is typically called the destination operand. The second operand is usually called the source operand.
- In general, the content s of the destination operand are modified . In the MOV instruction, data is copied from the source to the destination

Comments

- Comments are an important way for the writer of a program to communicate information about the program's design to a person reading the source code

example

Label	Mnemonic	Operand	Comment
START:	MVI	A, 22H	;Move 22_{16} to Register A
	ADI	20H	;Add 20_{16} to Register A contents
	LXI	H, 8400H	;Load register pair HL with 8400_{16}
	MOV	M, A	;Move contents of Reg A into location 8400_{16}
	JMP	START	;Jump to START

Instruction Types

- An instruction normally has two components, the operation code (op-code) and the data or address to be operated on (Operand).
- To perform an operation, operands must be given.
- Operands can come from registers, memory locations or directly **(Immediate**) as part of instructions.

Instruction Groups

- Generally instructions available in a microprocessor may be broadly classified into five groups
  - Data transfer
  - Arithmetic
  - Logical
  - Program control
  - Input/Output

Data Transfer Instructions

- Instructions for moving/transferring data between the processor and main memory.
  - Register to Register
  - Register to Memory
  - Memory to Register
  - Memory to Memory

Example

MOV B,C ; the instruction moves contents of register C to register B.

Arithmetic Instructions

- All instruction sets include ADD and SUBTRACT instructions. Some also include multiplication and division.
- To facilitate floating point number manipulations, some processors include floating-point instructions in their instructions set.
- Example ADD B; Add contents of register B to the contents in the accumulator.

Logical Instructions

- These includes instructions for performing the Boolean AND, NOT, OR, and EXCLUSIVE-OR operations on a bit-by-bit basis.
- Also included are SHIFT instructions:
  - Shift left or right
  - Rotate left or right

Program Control Instructions

- Instructions are always executed in the same order they are represented. In a real-life situation, flow of control depends on the results of computation.
- Based on the results of a computation, a program can select a particular sequence of instructions to execute.
- Instructions that realize this approach are called **program control instructions**.

cont

- Program Control instructions may be classified into the following groups:
  - Unconditional branch instructions
  - Conditional branch instructions
  - Subroutine call instructions

- **Unconditional Branch Instruction** transfers control to a specified address regardless of the status of a computation.
- **A conditional Branch instruction** transfers control when some condition is met after a computation.
  - **if**(condition) ***then*** branch to execute a new instruction ***else*** the instructions that follow.

Subroutine call.

- A subroutine is a special program for performing repeatedly needed tasks such as sorting, searching…
- In this case the control of the program will now change to execute the subroutine. Control returns after finishing with the subroutine.

Instructions set

- 8085 has a total of 246 instructions (see table)
- Familiarization with the instructions set

Comp 222

ASSEMBLY LANGUAGE PROGRAMING

- At the end of this course **Comp 222** you should be able to;
  - State what a microprocessor is, and discuss the function of the main parts in a microprocessor
  - Explain the difference between a microprocessor, a microcomputer and a microcontroller
  - Identify the function of the address, data, and control bus
  - Make comparison on the different programming languages i.e machine language, assembly language, and high level languages

- Trace the flow of data through the Internal parts of a microprocessor
- Familiarize with a microprocessor’s instructions set
- Write simple programs in Assembly Language

Stored-Program Computer

- One of the first digital computers was a machine called Electronic Numerical Integrator And Computer (ENIAC). First built at the Moore School of Electrical Engineering at the University of Pennsylvania in 1946.
- The computer measured 6M high, about 27M long and covered an area of 500 Square Metres
- The machine was programmed by setting up to 6000 switches and connecting cables between various units.

- While ENIAC was under construction, a Dr John Von Neumann also of Moore School of Electrical Engineering wrote a paper that would define the architecture to be used by nearly all computers from that day on.
- Neumann suggested that , rather than rewire the computer for each new task, the program instructions should be stored in a memory unit just like the data. The resulting computer was software programmable rather than hardware programmable.

- The Von Neumann‘s proposal is now called the stored programmed concept.
- There are three main parts of a stored-program computer.
  - CPU- Central Processing Unit
  - Memory Unit
  - Input/output Devices

Central Processing Unit(CPU)

- Where all decisions are made and the system timing signals are generated. The Arithmetic Logic Unit or ALU, is contained within the CPU and all mathematical operations are performed there. The results of these calculations are left in a special register in the ALU called the **Accumulator**.
- The CPU also contains the Control Unit that controls and coordinates the operation of other units (memory, input and output devices).

The memory Unit

- Used to store the specific sequence of commands that will be used to instruct the CPU to perform some task.
- These commands/instructions are called the computer program hence the name stored-program computer.
- Each cell in the memory has its own identifying number that is referred to as **address**

- These memory cells contain a strange collection of numbers having no particular meaning to us.
- To the CPU these numbers would represent a concise set of commands instructing it to carry out a given sequence of operations.
- These numbers represent the operation codes **(Op Codes)** for the various instructions in the CPU's instruction set.

The Input/Output Devices

- It is through the keyboard that we input the instructions or commands about the task to be accomplished. The results are then viewed on the printer, or Cathode Ray Tube (CRT) screen.

Fetch-Execute Cycle

- The CPU is designed to follow the following steps repeatedly

- **1.** Fetch data from a memory cell whose address is currently in the Program Counter(PC) Register and put the data into the Instruction Register(IR**) - (FETCH).**
- **2.** Add 1 to the address in the Program Counter i.e increment the program to point to the next memory cell.

- **3.** Decode the command that is currently in the Instruction Register and do what it says. **(Decode** and **Execute).**
- **4.** Go to step 1

- The steps are simplified to

  Fetch -> Decode -> Execute

The Three-Bus Architecture

- The CPU, Memory, and I/O devices must be able to communicate with each other.
- Such communication is facilitated by the **address, data** and **control buses.**
- Bus is a set of wires/cables used for address, data and control signals transmission.
- The CPU must specify which memory cell is to be selected and whether the content of that cell should be read or whether new data should be written into the cell **(Read/Write controls).**

- When reading a particular memory
  a. Address of the memory cell is placed on **Address bus**
  b. Read signal is placed on **Control Signal bus**
  c. Data from the addressed cell is placed on the **Data bus**

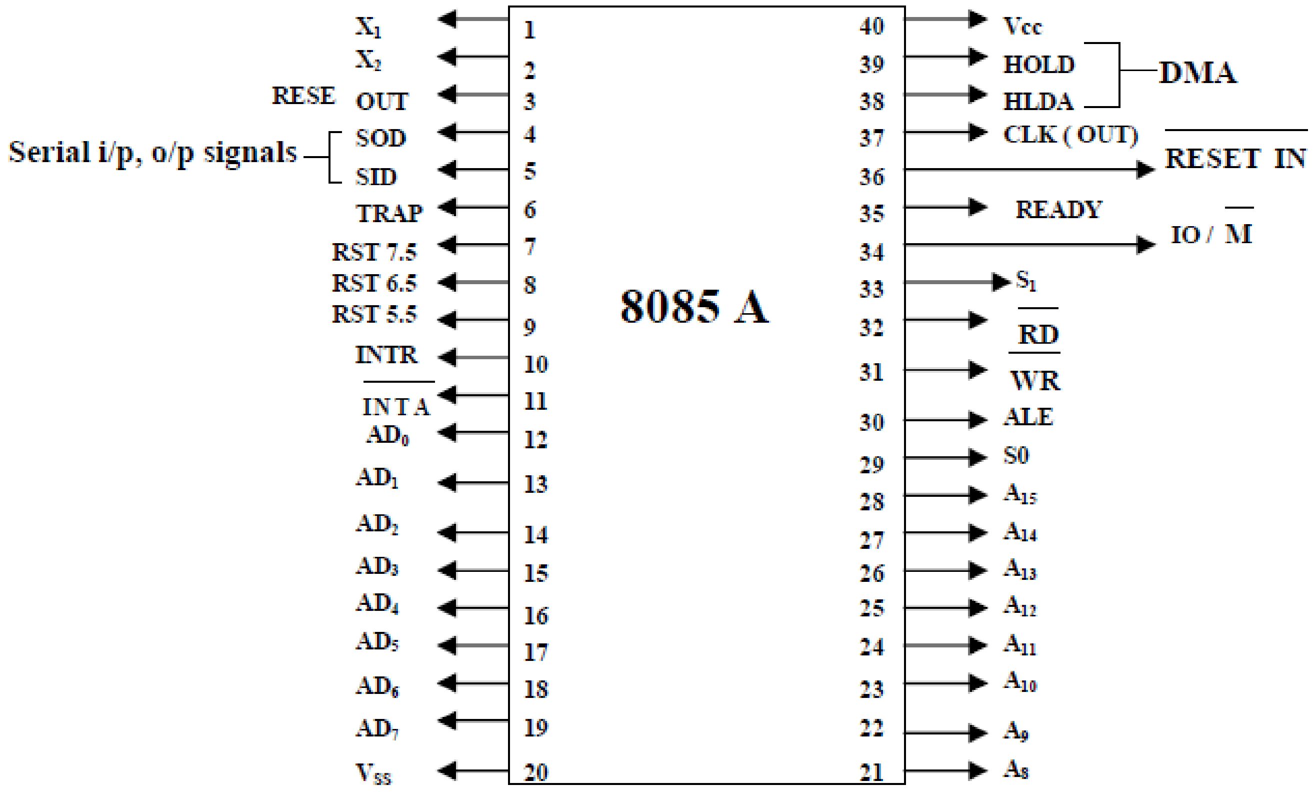
In summary, the CPU begins each command cycle with an instruction fetch from the memory unit. The program counter is then incremented in preparation for the next fetch. Finally the OpCode for the instruction is decoded and executed during the execution phase of the cycle.

INTEL 8085 MICROPROCESSOR

Introduction to Microprocessors

- A microprocessor is a general purpose device that is driven by a set of instructions and communicates with several external support chips to perform input/output of specific tasks.
- A microprocessor is also commonly referred to as CPU.
- New microprocessors come to the market every year to meet the needs of designers, however the theory behind this technology remains basically the same.
- This course will be based on the 8085 microprocessor that runs at a frequency of either 3.03MHZ(8085A) or 5MHZ maximum(8085A-2).

An 8085A Microprocessor



Pin Diagram of 8085

Signal Groups of 8085

Pin Arrangement

- The pins on the chip can be grouped into six groups:
  - Address Bus
  - Data Bus
  - Power supply and clock signals
  - Serial I/O ports(SID,SOD)
  - Control and Status signals
  - Externally initiated signals

Address and Data Bus

- Address bus has 8 lines $A_8 – A_{15}$ which are unidirectional. The remaining 8 bits address are multiplexed (time shared) with the 8 data bit lines.
- $AD_0 – AD_7$ serve both as bi-directional data bus $D_0 – D_7$ and $A_0 – A_7$ for address lines to make 16-bits with $A_8 – A_{15}$
- During the execution of the instruction, the lines will carry the address bits and during execution will carry the 8 bit data.
- Latches can be used to save/hold the information before the function of the bits change.

Clock signals

- **X_1** and **X_2** are the inputs from the crystal oscillator or clock generating circuit.

- **Clk out** pin: An output clocking pin to drive the rest of the system.

Control and Status signals

- **ALE: Address Latch Enable**. Set to 1 by the system when $AD_0 – AD_7$ lines have an address on them. It changes back to 0 thereafter. This signal can be used to save the address bits from the AD lines.
- **RD: Read.** Active Low signal to enable a read from a memory location.
- **WR: Write.** Active Low signal to enable a write to a memory location.

Cont.

- **IO/M**:Input,Output/Memory. This signal specifies whether the operation is a memory operation(IO/M = 0) or an I/O operation (IO/M =1).
- **S_1** and **S_2** : Status signals to specify the kind of operation being performed.

Externally Initiated Signals(Interrupts)

- **What is an interrupt? RST, TRAP** to be covered later under interrupts.
- Interrupt signals can externally or internally generated.
- E.g when attempting to divide a number by zero or accessing a part of memory that is not allowed.
- When an external device wants to be served by the CPU through the input/output ports.

8085 Features

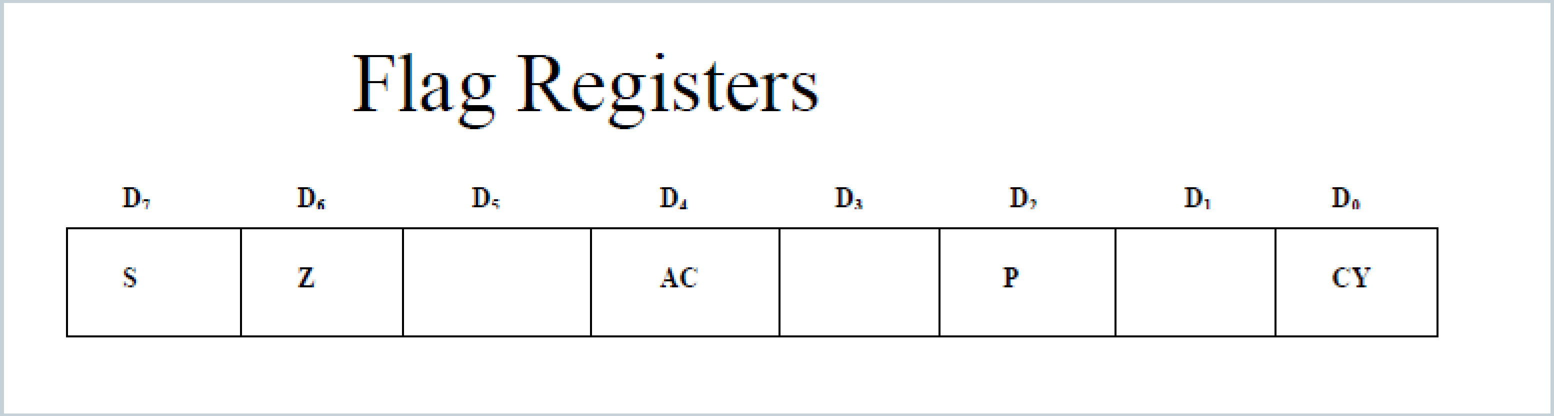
- **The ALU**
  - In addition to the arithmetic & Logic circuits, the ALU includes the accumulator, which is part of every arithmetic and logic operation.
  - The ALU also includes a temporary register that is used for temporary data storage during the execution of an operation. The temporary register is not accessible by the programmer.

Cont.

- **The Flag Register(program status word)**
  - The flag register consists of five status flags
    - **The sign flag(S)**: This is set to the value of the most significant bit of the accumulator after an arithmetic or logic operation (0 for positive and 1 for negative).
    - **The zero flag(Z)** is set to a 1 whenever an arithmetic or logic operation produces a result of zero. A nonzero result sets it to 0.

Cont.

- **The auxiliary carry flag(Ac)**: Reflects any carry from bit 3 to bit 4(assuming an 8-bit data with bit 0 as the LSB and bit 7 as the MSB).
- **The parity status flag(P)**: is set to 1 if an operation produces an answer with even parity (even number of 1's e.g 11100001) or set to 0 if the answer has odd number of 1's.
- **The carry flag(Cy):** Reflects the final carry out of the most significant bit of any arithmetic operation. The flag is also used for the shift and rotate instructions.



Flag Registers

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z		AC		P		CY

Register Structure

The Accumulator:

- The accumulator(A) is an 8-bit register.
- Most arithmetic and logic operations are performed using the accumulator.
- All I/O operations are performed via the accumulator

Cont.

General purpose registers:

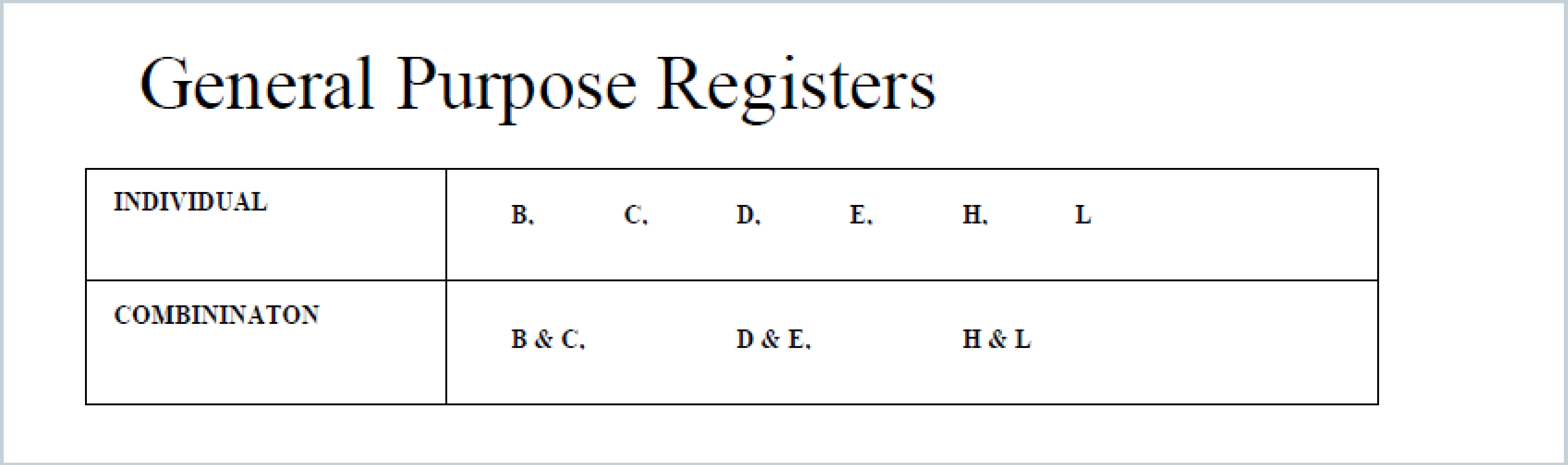
- The B,C,D,E,H and L registers are 8-bits long and are used for moving data between themselves, the accumulator and the memory.

- There are a number of instructions that combine two of these 8-bit registers to form 16-bit register pair.

Cont.

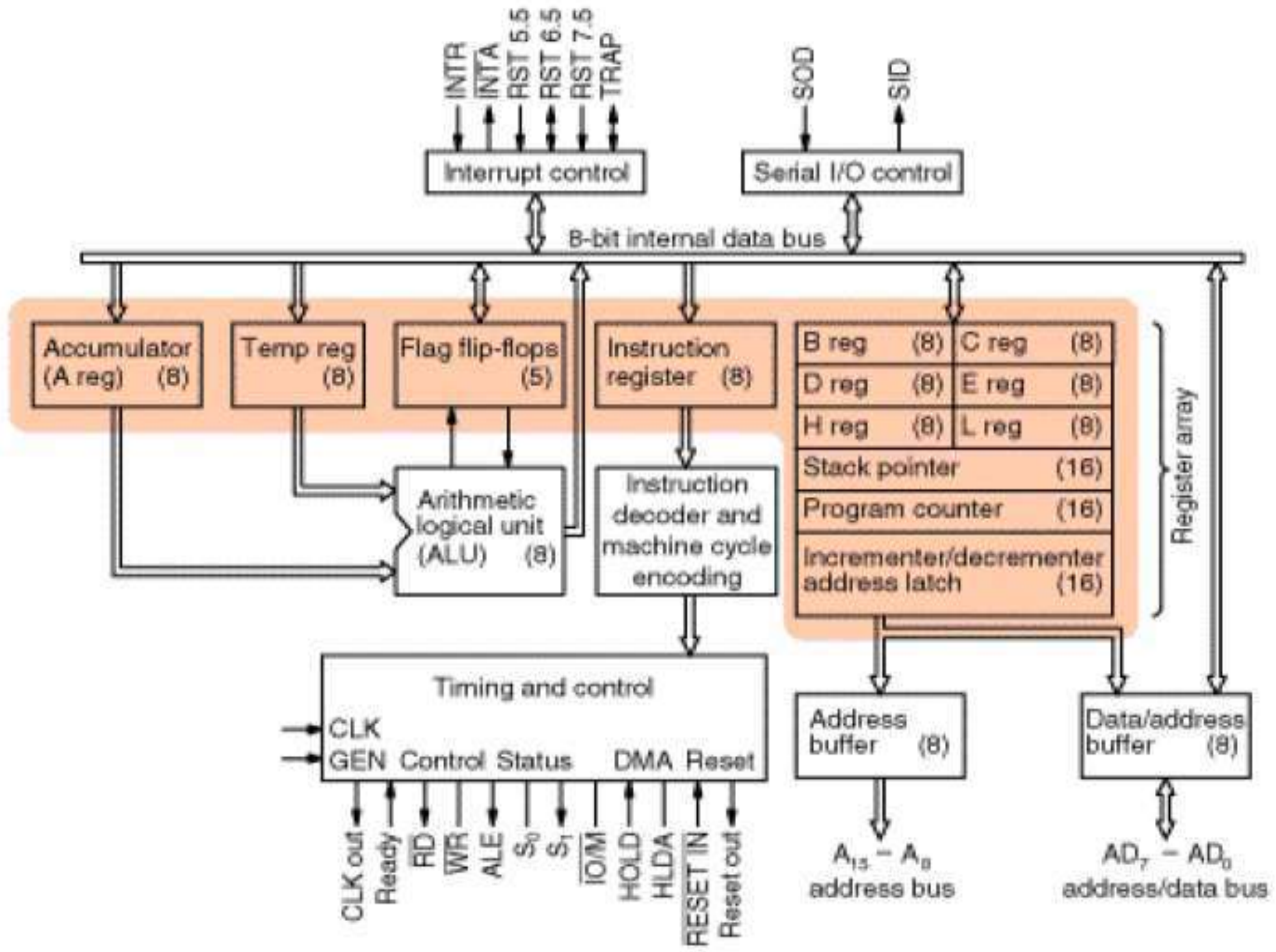
- Arithmetic operations use B and C, or D and E, or H and L as 16-bit data registers.

- Register HL is the memory address register or data counter. The register pair stores the 16-bit address of an 8-bit data being accessed from memory.



General Purpose Registers

INDIVIDUAL	B,	C,	D,	E,	H,	L
COMBININATON	B & C,		D & E,		H & L	



INTR
INTA
RST 5.5
RST 6.5
RST 7.5
TRAP
SOD
SID
Interrupt control
Serial I/O control
8-bit internal data bus
Accumulator (A reg) (8)
Temp reg (8)
Flag flip-flops (5)
Instruction register (8)
B reg (8)
C reg (8)
D reg (8)
E reg (8)
H reg (8)
L reg (8)
Stack pointer (16)
Program counter (16)
Incrementer/decrementer address latch (16)
Register array
Arithmetic logical unit (ALU) (8)
Instruction decoder and machine cycle encoding
Timing and control
CLK GEN
Control
Status
DMA
Reset
CLK out
Ready
RD
WR
ALE
S0
S1
IO/M
HOLD
HLDA
RESET IN
Reset out
Address buffer (8)
Data/address buffer (8)
A15 – A8 address bus
AD7 – AD0 address/data bus

Fetching an Instruction

- Assume the processor is fetching an instruction from address 8400H. That means the Program Counter(PC) is loaded with this value.
  - The PC places the address value on the address bus and the controller issues a RD signal.
  - The memory's address decoder gets the value and determines which memory location is being accessed.

Cont.

- The value on the memory location is placed on the data bus.
- The value on the data bus is read into the Instruction Decoder(ID) inside the microprocessor.
- After decoding the instruction, the control unit issues control signals to perform the operation.