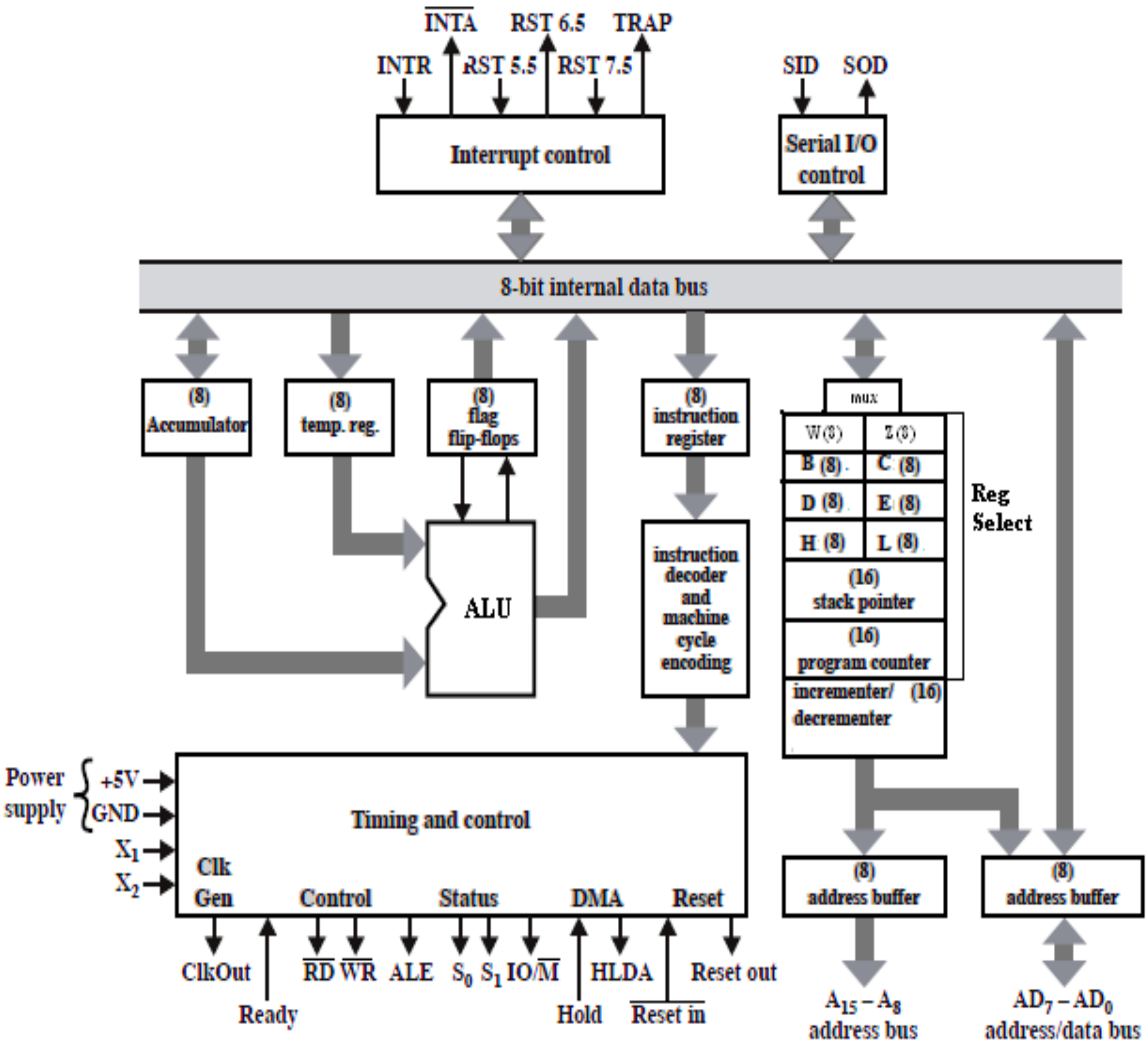
8085 ARCHITECTURE, PINS AND FLAG REGISTER

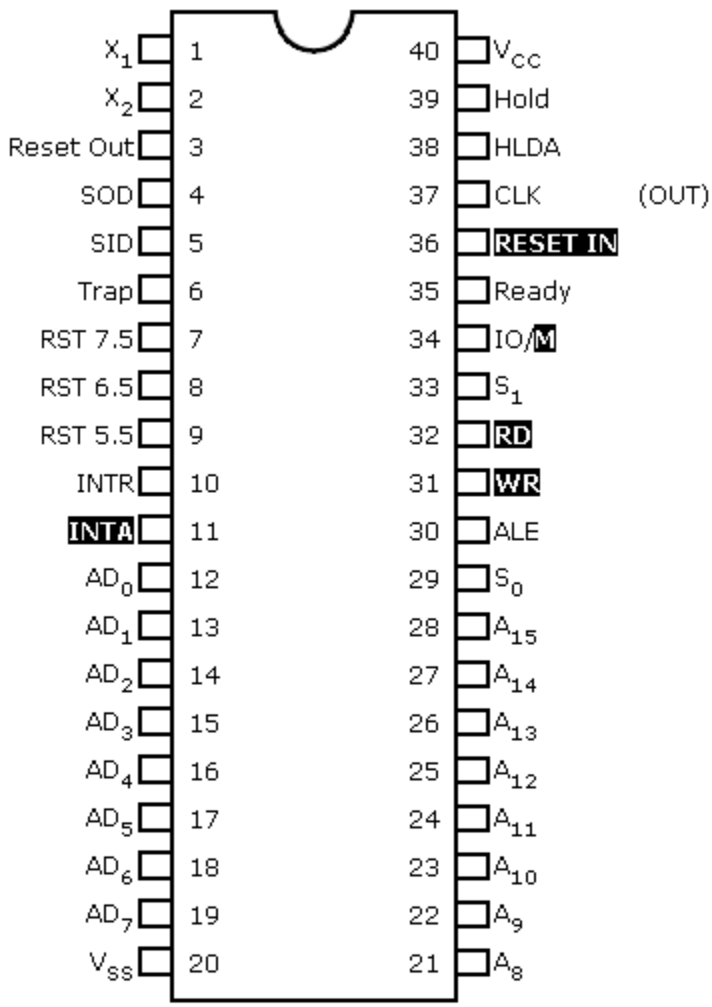
Note

Dear Students, Architecture contains all the pins and the Flag Register.
Hence, I have made a common PDF for all these three topics.

8085 ARCHITECTURE



8085 PIN DIAGRAM



REGISTERS

Program Counter (PC, 16-bits)

It is a 16-bit Special-Purpose register. It holds **address** of the **next instruction**.
PC is incremented by the INR/DCR after every instruction byte is fetched.

Stack Pointer (SP, 16-bits)

It is a 16-bit Special-Purpose register. It holds **address** of the **top of the Stack**.
Stack is a set of memory locations operating in LIFO manner.
SP is **decremented** on every **PUSH** operation and **incremented** on every **POP**.

B, C, D, E, H, L (8-bits)

These are 8-bit General-Purpose registers.
They can also be used to store 16-bit data in register pairs.
The possible register **pairs** are **BC** pair, **DE** pair and **HL** pair.
The **HL** pair also holds the **address** for the Memory Pointer **"M"**.

Temporary Register Pair (WZ, 16-bits)

This is a 16-bit register pair.
It is **used by µP** to hold **temporary** values in some instructions like CALL/JMP/LDA etc.
The **programmer** has **no access** to this register pair.

INR/DCR Register (16-bits)

This is a 16-bit shift register.
It is used to **increment PC after every instruction byte is fetched**
It also **increments** or **decrements SP** after a Pop or a Push operation respectively.
It is not available to the programmer.

A - Accumulator (8-bits)

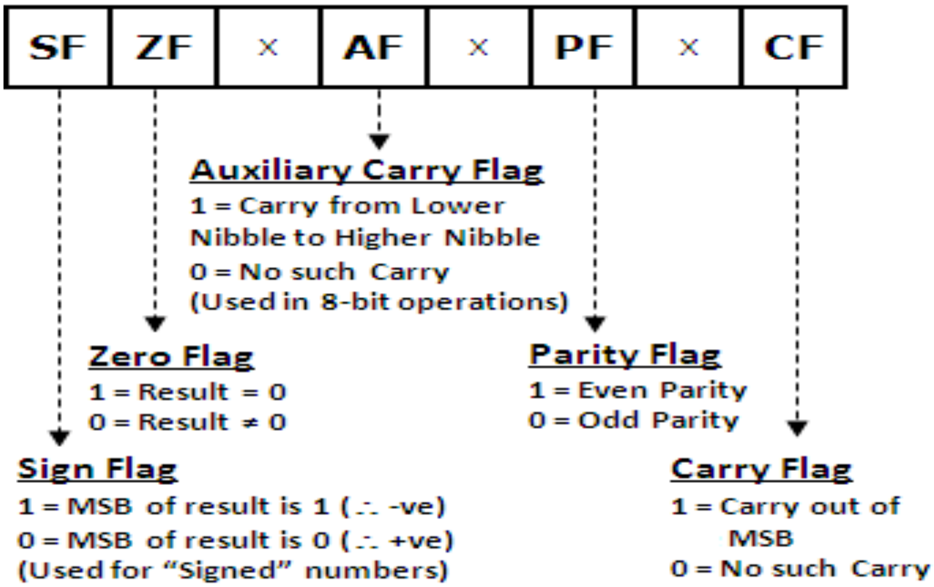
It is an 8-bit programmable register.
The user can read or write this register.
It has two **special properties** viz:

- It **holds the first operand** during most arithmetic operations.
- It **holds** the **result** of most of the arithmetic and logic operations
  Eg: ADD B; This instruction will do A + B and store the result in A.
  Eg: SUB B; This instruction will do A - B and store the result in A.

Temp Register (8-bits)

This is an 8-bit register.
It is **used by μP** for storing one of the operands during an operation.
The **programmer** has **NO ACCESS** to this register.

8085 FLAG REGISTER



S - Sign Flag

It is **set** (1) when **MSB** of the result is **1** (i.e. result is a **-**VE number).
It is reset (0) when MSB of the result is 0 (i.e. result is a **+**VE number).

Z - Zero Flag

It is **set** when the **result** is **= zero**.
It is reset when the result is not = zero.

AC - Auxiliary Carry Flag

It is **set** when an **Auxiliary Carry** / Borrow is **generated**.
It is reset when an Auxiliary Carry / Borrow is not generated.
Auxiliary Carry is the Carry generated **between** the **lower nibble and the higher nibble** for an 8-bit operation. It is not affected after a 16- bit operation. It is used only in DAA operation.

P - Parity Flag

It is **set (1)** when result has **even parity**. It is reset when result has odd parity.

C - Carry Flag

It is **set** when a **Carry / Borrow** is **generated from the MSB**.
It is reset when a Carry / Borrow is not generated from the MSB.

 # In the exam, Show at least 2 examples from Bharat Sir's video lecture

INTERRUPT CONTROL

This Block is responsible for controlling the **hardware interrupts** of 8085.
8085 supports the following hardware interrupts:

TRAP

This is an **edge as well as level triggered**, **vectored** interrupt.
It cannot be masked by SIM instruction and can neither be disabled by DI instruction.
It has the **highest priority**.
Its vector address is **0024H**.

RST 7.5

This is an **edge triggered**, **vectored** interrupt.
It can be masked by SIM instruction and can also be disabled by DI instruction.
It has the **second highest priority**.
Its vector address is **003CH**.

RST 6.5

This is a **level triggered**, **vectored** interrupt.
It can be masked by SIM instruction and can also be disabled by DI instruction.
It has the **third highest priority**.
Its vector address is **0034H**.

RST 5.5

This is a **level triggered**, **vectored** interrupt.
It can be masked by SIM instruction and can also be disabled by DI instruction.
It has the **fourth highest priority**.
Its vector address is **002CH**.

INTR

This is a **level triggered**, **non-vectored** interrupt.
It cannot be masked by SIM instruction but can be disabled by DI instruction.
It has the **lowest priority**.
It has an **acknowledgement signal** $\overline{\textbf{INTA}}$.
The address for the ISR is **fetched from external hardware**.

$\overline{\textbf{INTA}}$

This is an **acknowledgement** signal **for INTR** (only).
This signal is used to **get** the Op-Code (and hence the ISR address) from External hardware in order to execute the ISR. ☺ In case of doubts, contact Bharat Sir: - 98204 08217.

ALL Interrupts **EXCEPT TRAP** can be **disabled** though the **DI** instruction.
These interrupts can be **enabled** again by the **EI** Instruction.
Interrupts can be individually **masked or unmasked by SIM instruction**.
TRAP and INTR are not affected by SIM instruction.

SERIAL CONTROL

This Block is responsible for transferring data Serially to and from the µP.

SID - Serial In Data

µP receives data, bit-by-bit through this line.

SOD - Serial Out Data

µP sends out data, bit-by-bit through this line.
Serial transmission can be done by **RIM** and **SIM** Instructions.

ALU

8085 has an **8-bit ALU**.
It performs 8-bit arithmetic operations like Addition and Subtraction.
It also performs logical operations like AND, OR, EX-OR NOT etc.
It takes **input** from the **Accumulator** and the **Temp** register.
The **output** of most of the ALU operations is stored back **into** the **Accumulator**.

INSTRUCTION REGISTER AND DECODER

Instruction Register

The 8085 places the contents of the PC onto the Address bus and fetches the instruction.
This fetched instruction is stored into the Instruction register.

Instruction Decoder:

The fetched instruction from the Instruction register enters the Instruction Decoder. Here the instruction is decoded and the decode information is given to the Timing and Control Circuit where the instruction is executed.

TIMING AND CONTROL CIRCUIT

The timing and control circuit issues the various internal and external control signals for executing and instruction.
The external pins connected to this circuit are as follows:

X1 and X2

These pins provide the **Clock Input to the µP**.
Clock is provided from a crystal oscillator.

ClkOut

8085 provides the **Clock input** to all **other peripherals** through the ClockOut pin.
This takes care of **synchronizing** all peripherals with 8085.

$\overline{\text{ResetIn}}$

This is an active low signal activated when the manual reset signal is applied to the µP. This signal **resets the µP**. On Reset PC contains **0000H**. Hence, the **Reset Vector Address** of 8085 is 0000H.

ResetOut

This signal is connected to the reset input of all the peripherals.
It is used to **reset the peripherals once** the **µP** is **reset**.

READY

This is an active high input.
It is used to **synchronize** the μP with **"Slower"** Peripherals.
The **μP samples** the **Ready** input in the beginning of every Machine Cycle.
If it is found to be **LOW**, the **μP executes** one **WAIT CYCLE** after which it re-samples the ready pin till it finds the Ready pin HIGH.
∴ The **μP remains** in the **WAIT STATE until** the **READY** pin becomes **high** again.
Hence, **if** the **Ready** pin is **not required** it should be **connected** to the **Vcc,** and not, left unconnected, **otherwise** would cause the μP to execute **infinite wait cycles**.
#Please refer Bharat Sir's video Lecture for this ...

ALE - Address Latch Enable

This signal is **used to latch address** from the multiplexed Address-Data Bus (**AD0-AD7**). When the Bus contains **address**, **ALE** is **high**, **else** it is **low**.

IO/ $\overline{M}$

This signal is used to distinguish between an **IO** and a **Memory operation**.
When this signal is high it is an IO operation else it is a Memory operation.

$\overline{RD}$

This is an active low signal used to indicate a **read operation**.

$\overline{WR}$

This is an active low signal used to indicate a **write operation**.

S_1 and S_0

These lines denote the status of the μP

S_1 S_0	Status
0 0	Idle
0 1	Write
1 0	Read
1 1	Opcode fetch

HOLD and HLDA

The Hold and Hold Acknowledge signals are used for **Direct Memory Access** (DMA).
The **DMA Controller issued** the **Hold** signal to the µP.
In response the **µP releases** the **System bus**.
After releasing the system bus the **µP acknowledges** the Hold signal with **HLDA** signal. The **DMA Transfer** thus **begins**.
DMA **Transfer** is **terminated** by **releasing** the **HOLD** signal.