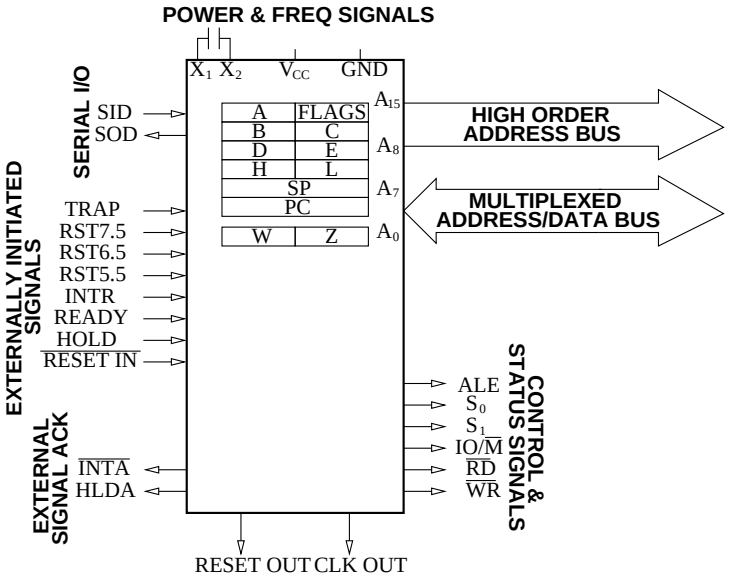
EE309: Computer Organization, Architecture and MicroProcessors

http://www.ee.iitb.ac.in/∼sumantra/courses/up/up.html

The 8085 Chip



FLAGS: S Z x A x P x cy

ALE: Address Latch Enable: Positive-going pulse at the start of a machine cycle (each 8085 operation), indicates AD_7 - AD_0 are A_7 - A_0.

External devices or signals can initiate the following four operations:

1. **Reset:**
   $\overline{RESET\ IN}$, $RESET\ OUT$
   Resets the 8085
   $PC \leftarrow 0$
2. **Ready:**
   The 8085 waits for an integral number of cycles till this goes high. It waits to sync with slow peripherals
3. **Hold:**
   $HOLD$, $HLDA$
   (DMA). The 8085 relinquishes the control of buses, allows external peripherals to use them
4. **Interrupt:**
   (a) $TRAP$: Non-maskable interrupt, highest priority
   (b) $RST7.5$, $RST6.5$, $RST5.5$: Vectored restart interrupts (transfer program control to specific memory locations). Order of priority: $7.5 > 6.5 > 5.5$
   (c) $INTR$: Software interrupt, general purpose
   (d) $INTA$: Interrupt Acknowledge

Machine Cycle	**(Status)** $IO/\overline{M}$	S_1	S_0	**Control Signals**
Opcode Fetch	0	1	1	$\overline{RD} = 0$
Memory Read	0	1	0	$\overline{RD} = 0$
Memory Write	0	0	1	$\overline{WR} = 0$
I/O Read	1	1	0	$\overline{RD} = 0$
I/O Write	1	0	1	$\overline{WR} = 0$
Interurpt Ack	1	1	1	$\overline{INTA} = 0$
Halt	Z	0	0	$\overline{RD} = Z$,
Hold	Z	X	X	$\overline{WR} = Z$,
Reset	Z	X	X	$\overline{INTA} = 1$

The 8085 Instruction Set: Programmer's View

1. Data Transfer Group
2. Arithmetic Group
3. Logical Group
4. Branch Group
5. Stack, I/O and Machine Control Instructions

1. Data Transfer Group
   `MOV, MVI, LXI, LDA, STA, LHLD, SHLD, LDAX, STAX, XCHG`

2. Arithmetic Group
   `ADD, ADI, ADC, ACI, INR, INX, DAD, DAA`
   `SUB, SUI, SBB, SBI, DEC, DCX`

3. Logical Group
   `ANA, ORA, XRA, CMP, RLC, RAL, CMA, STC`
   `ANI, ORI, XRI, CPI, RRC, RAR, CMC`

4. Branch Group

```
JMP,      JC,  JZ,  JP, JPE,     CALL,      CC,  CP, CPE,  CZ
        JNC, JNZ,  JM, JPO,      RET,     CNC,  CM, CPO, CNZ
                                           RC,  RP, RPE,  RZ
RST,            PCHL,                     RNC,  RM, RPO, RNZ
```

5. Stack, I/O and Machine Control Instructions

```
PUSH, XTHL,  IN, EI,   HLT, NOP,   SIM,   LXI SP,   INX SP,   DAD SP
 POP, SPHL, OUT, DI,               RIM,            DCX SP
```

1. Data Transfer Group [No Flags Affected]

1. `MOV` r_1, r_2
   $r_1 \leftarrow r_2$

2. `MOV` r, `M`
   $r \leftarrow (HL)$
   The contents of the memory location stored in the HL pair are copied to register r

3. `MOV M`, r
   $(HL) \leftarrow r$

4. `MVI` r, 8-*bit data*
   $r \leftarrow byte\_2$

5. `MVI M`, 8-*bit data*
   $(HL) \leftarrow byte\_2$
   Transfer the byte to the memory location given by the contents of the HL pair

6. `LXI` *rp*, 16-*bit data*
   $rh \leftarrow byte\_3$
   $rl \leftarrow byte\_2$
   `LXI B`, `LXI D`, `LXI H` load two bytes into the register pairs BC, DE and HL, respectively. The 8085 convention: the lower-byte of a 2-byte pair is stored first in memory

7. `LDA` 16-*bit address*
   $A \leftarrow (byte\_3\ byte\_2)$

8. `STA` 16-*bit address*
   $(byte\_3\ byte\_2) \leftarrow A$

9. `LHLD` 16-*bit address*
   $L \leftarrow (byte\_3\ byte\_2)$
   $H \leftarrow (byte\_3\ byte\_2\ +\ 1)$

10. `SHLD` 16-*bit address*
    $(byte\_3\ byte\_2) \leftarrow L$
    $(byte\_3\ byte\_2\ +\ 1) \leftarrow H$

11. `LDAX` *rp*
    $A \leftarrow (rp)$, *i.e.*, $A \leftarrow (rh\ rl)$
    The Accumulator is loaded with the contents of the address (2-byte value) stored in the register pair. NOTE: This is not valid for the HL pair !

12. `STAX` *rp*
    $(rp) \leftarrow A$, *i.e.*, $(rh\ rl) \leftarrow A$

13. `XCHG`
    $H \leftrightarrow D$
    $L \leftrightarrow E$

2. Arithmetic Group

[All Flags Affected:]

1. `ADD` r
   $A \leftarrow A + r$
   NOTE: `ADD A` is a valid instruction !
2. `ADD M`
   $A \leftarrow A + (HL)$
3. `ADI` 8-*bit data*
   $A \leftarrow A + byte\_2$
4. `ADC` r
   $A \leftarrow A + r + cy$
5. `ADC M`
   $A \leftarrow A + (HL) + cy$
6. `ACI` 8-*bit data*
   $A \leftarrow A + byte\_2 + cy$
7. `SUB` r
   $A \leftarrow A - r$
   If the result is negative, the Carry / Borrow flag cy is set. The Carry flag is the Borrow flag. However, the Auxiliary Carry flag does not double up as an Auxiliary Borrow !
8. `SUB M`
   $A \leftarrow A - (HL)$
9. `SUI` 8-*bit data*
   $A \leftarrow A - byte\_2$
10. `SBB` r
    $A \leftarrow A - \{r + cy\}$
11. `SBB M`
    $A \leftarrow A - \{(HL) + cy\}$
12. `SBI` 8-*bit data*
    $A \leftarrow A - \{byte\_2 + cy\}$
13. `DAA`
    This adjusts the accumulator to packed BCD after the addition of two BCDs. It functions in two steps:
    (a) If the lower 4 bits of A are greater than 9, *or* the Auxiliary Carry flag is set, then it adds 6 to the lower nibble of A
    (b) Subsequently, if the higher 4 bits of A are now greater than 9, *or* the Carry Flag is set, it adds 6 to the higher nibble of A

[Only cy Flag Affected:]

14. `DAD` rp
    $HL \leftarrow HL + rp$

[Flags Affected: $ZAPS$, no cy !]

15. `INR` r
    $r \leftarrow r + 1$

16. `INR M`
    $(HL) \leftarrow (HL) + 1$

17. `DCR` r
    $r \leftarrow r - 1$

18. `DCR M`
    $(HL) \leftarrow (HL) - 1$

[No Flags Affected:]

19. `INX` rp
    $rp \leftarrow rp + 1$ $\qquad$ *i.e.*, $rh\ rl \leftarrow rh\ rl + 1$

20. `DCX` rp
    $rp \leftarrow rp - 1$ $\qquad$ *i.e.*, $rh\ rl \leftarrow rh\ rl - 1$

Addition and Subtraction + `DAA` $\longmapsto$ All Flags !
`INR` + `DCR` $\longmapsto$ no cy !
`DAD` $\longmapsto$ only cy !
`INX` + `DCX` $\longmapsto$ no flags !

3. Logical Group

1. `ANA` r
   $A \leftarrow A \wedge r$

2. `ANA M`
   $A \leftarrow A \wedge (HL)$

3. `ANI` 8-*bit data*
   $A \leftarrow A \wedge byte\_2$

4. `ORA` r
   $A \leftarrow A \vee r$

5. `ORA M`
   $A \leftarrow A \vee (HL)$

6. `ORI` 8-*bit data*
   $A \leftarrow A \vee byte\_2$

7. `XRA` r
   $A \leftarrow A \oplus r$

8. `XRA M`
   $A \leftarrow A \oplus (HL)$

9. `XRI` 8-*bit data*
   $A \leftarrow A \oplus byte\_2$

10. `CMP` r
    $A - r$, without performing the actual subtraction. This just sets the flags.
11. `CMP M`
12. `CPI` 8-*bit data*
13. `RLC`
    Rotate left *without* Carry !
14. `RAL`
    Rotate Accumulator left through Carry
15. `RRC`
    Rotate right *without* Carry !
16. `RAR`
    Rotate Accumulator right through Carry
17. `CMA`
    Complement contents of the Accumulator
18. `STC`
    Set the Carry flag
19. `CMC`
    Complement the Carry flag

AND, OR, XOR $\longmapsto$ $cyZPS$ affected in all, cy reset(0) AND $\longmapsto$ Auxiliary Carry is set(1) any OR (inclusive / exclusive) $\longmapsto$ Auxiliary Carry is reset(0)
Comparison $\longmapsto$ All flags
Rotation $\longmapsto$ only cy !
`CMA` $\longmapsto$ No flags !
`STC`, `CMC` $\longmapsto$ only cy !

4. Branch Group [No Flags affected !]

1. `JMP` 16-*bit address*
2. `JC` 16-*bit address*
3. `JNC` 16-*bit address*
4. `JZ` 16-*bit address*
5. `JNZ` 16-*bit address*
6. `JP` 16-*bit address*
7. `JM` 16-*bit address*
8. `JPE` 16-*bit address*
9. `JPO` 16-*bit address*
10. `CALL` 16-*bit address*
11. `CC` 16-*bit address*
12. `CNC` 16-*bit address*
13. `CZ` 16-*bit address*
14. `CNZ` 16-*bit address*
15. `CP` 16-*bit address*
16. `CM` 16-*bit address*
17. `CPE` 16-*bit address*
18. `CPO` 16-*bit address*
19. `RET` 16-*bit address*
20. `RC` 16-*bit address*
21. `RNC` 16-*bit address*
22. `RZ` 16-*bit address*
23. `RNZ` 16-*bit address*
24. `RP` 16-*bit address*
25. `RM` 16-*bit address*
26. `RPE` 16-*bit address*
27. `RPO` 16-*bit address*
28. `RST` n
    $0 \leq n \leq 7$. Restart $\rightarrow$ control is transferred to address $n * 8$.
29. `PCHL`
    $PCh \leftarrow H$
    $PCl \leftarrow L$

5. Stack, I/O and Machine Control Instructions

[No Flags affected unless explicitly mentioned]

1. `PUSH` *rp*
   Push the register pair contents onto the stack, with the higher-order byte pushed first

2. `POP` *rp*
   The low and then the high byte are popped from the stack onto the register pair

3. `PUSH PSW`
   The Program Status Word comprises the Accumulator (higher-order byte), and the Flags register (lower-order byte)

4. `POP PSW` [All Flags affected !]

5. `XTHL`
   $L \leftrightarrow top - of - stack$
   $H \leftrightarrow next - in - line$
   The contents of the HL pair are exchanged with the top of the stack (the lower-order byte is always on top). The contents of SP are not affected

6. `SPHL`
   $SPh \leftarrow H$
   $SPl \leftarrow L$
   The contents of the HL pair are *copied* onto the Stack Pointer register

7. `IN` 1-*byte port address*
   The port address is duplicated on both the higher-order ($A_{15} - A_8$) and lower-order ($AD_7 - AD_0$) address bus

8. `OUT` 1-*byte port address*

9. `EI`
   The Interrupt Enable Flip Flop is set. and all intrrupts are enabled (the TRAP interrupt is always enabled)

10. `DI`
    The Interrupt Enable Flip Flop is reset, and all interrupts are disabled (except the TRAP interrupt, which cannot be disabled

11. `HLT`
    Halt and enter Wait State, till an interrupt, or reset

12. `NOP`
    No Operation, used to insert precise time delays

13. `LXI SP` 16-*bit address*

14. `INX SP`

15. `DCX SP`

16. `DAD SP` [Only *cy* Flag affected]
    $HL \leftarrow HL + SP$

17. `SIM`
    Set Interrupt Mask A suitable bit pattern is loaded into the Accumulator, and then this instruction is to be called. The bit pattern is interpretated as follows: If SDE = 1 only (Serial Data Enable), then whatever is in the

SOD	SDE	×	R7.5	MSE	M7.5	M6.5	M5.5

    SOD (Serial Output Data) - 0/1 → is latched onto the SOD output line on the 8085.
    R7.5 is an additional control to reset the RST7.5 flip flop.
    If MSE = 1 only (Mask Set Enable), Mn is masked or disabled if Mn = 1, enabled if Mn = 0.
    Ths command is also used for serial output.

18. `RIM`
    Read Interrupt Mask This command is used to read the status of interrupts 7.5, 6.5 and 5.5, and serial data input. The result of executing this command is a number being loaded into the Accumulator, whose interpretation is as follows: SID represents the serial input bit.

SID	I7.5	I6.5	I5.5	IE	M7.5	M6.5	M5.5

    In = 1 ⇒ interrupt n is pending.
    IE = 1 ⇒ the Interrupt Enable flip flop is set.
    Mn ⇒ the corresponding interrupt is masked out.

Notes

"All are equal, but some are more equal than others."
. George Orwell, *Animal Farm.*

- HOLD has a higher priority than any interrupt.
- `LDAX` is not meant for the HL register pair.
- `DAD, HLT` have a special 'Bus Idle' Machine cycle.
- The following instructions have a special 6 T-state Opcode Fetch machine cycle: `INX, DCX, PCHL, SPHL, PUSH, RST`, and All CALL and RETURN instructions (except `RET`)