Bùi Minh Thành
Hiệu đính từ bài giảng của
thầy Hồ Trung Mỹ (BMDT- DHBK)

Chương 2

KIẾN TRÚC CPU VÀ TẬP LỆNH

Nội dung

2.1 Sơ đồ khối CPU 8 bit cơ bản

2.2 Tổ chức các thanh ghi

2.3 Tổ chức bộ nhớ

2.4 Ghép nối bus hệ thống

2.5 Chu kỳ bus, chu kỳ máy

2.6 Các phương pháp định địa chỉ

2.7 Tập lệnh

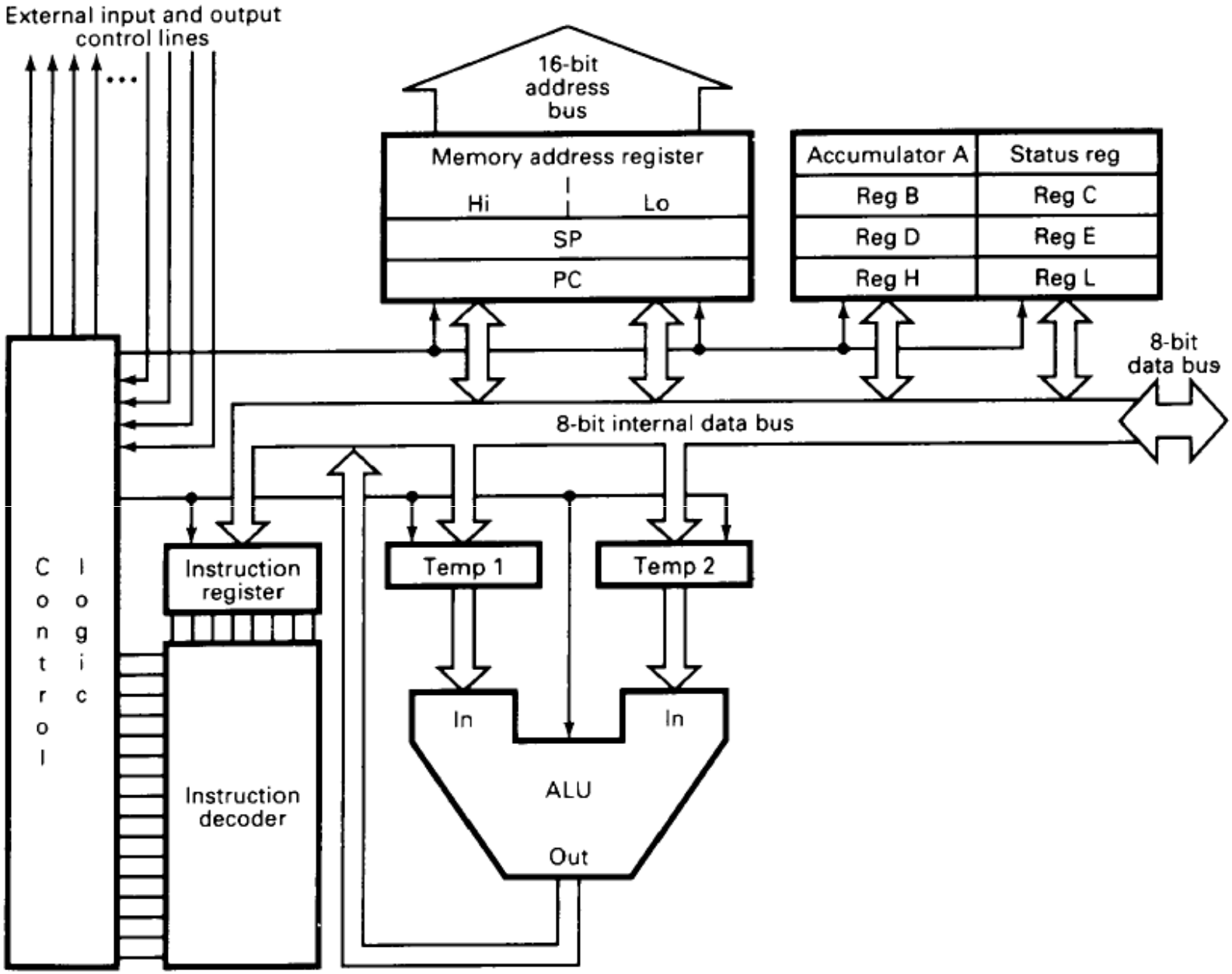
Vi xử lý

- Có nhiều loại vi xử lý (VXL) từ rất đơn giản đến rất phức tạp
- Phụ thuộc vào độ rộng bus dữ liệu và thanh ghi và ALU, có các VXL 4 bit , 8 bit , 16bit, 32 bit , 64 bit …
- Thí dụ
  - Z80 là VXL 8 bit
  - 8086/88 là VXL 16 bit
- Tất cả các VXL có
  - Bus địa chỉ
  - Bus dữ liệu
  - Các tín hiệu điều khiển: RD, WR, CLK , RST, INT, . . .

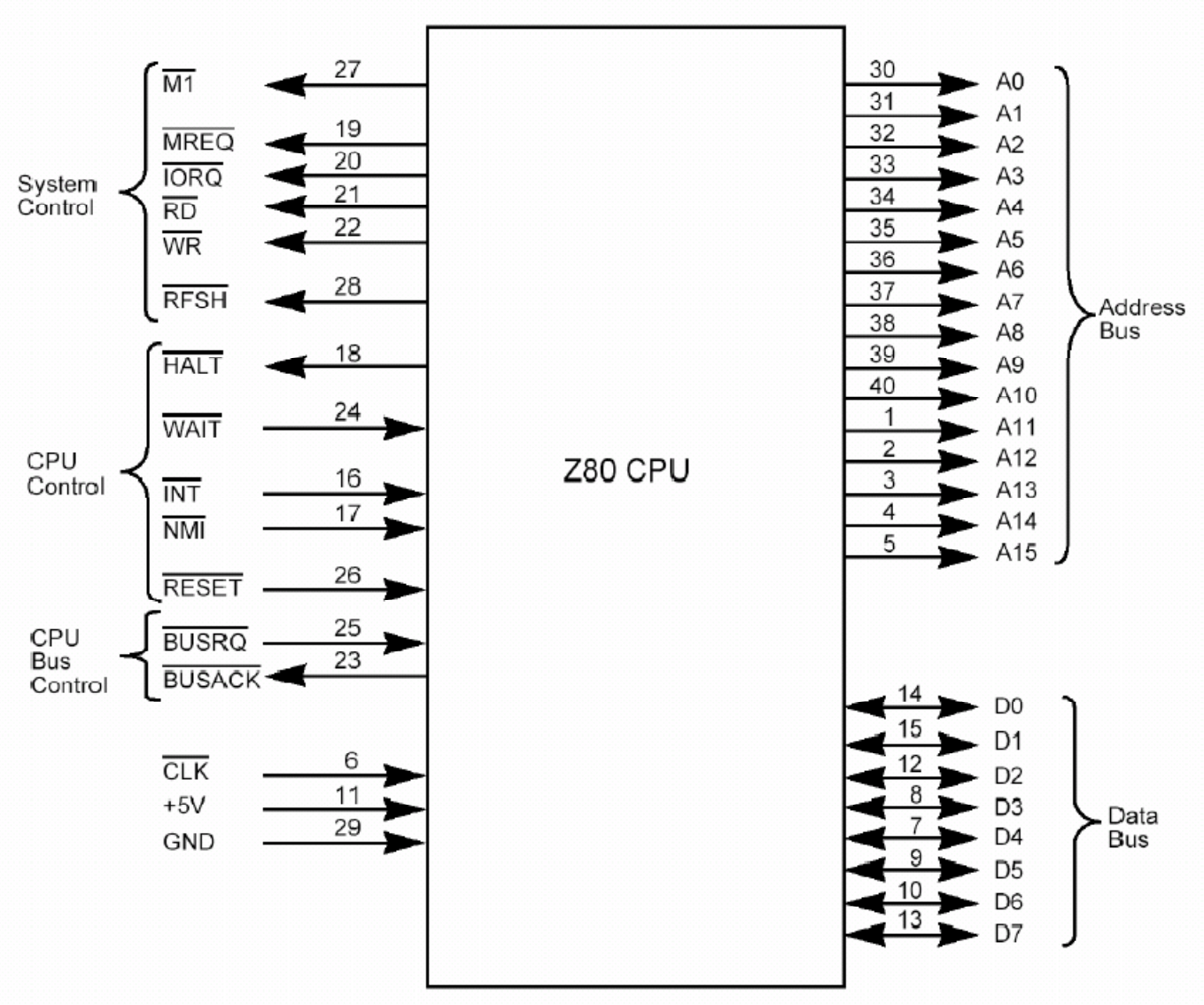
Bus nội và ngoại

- Bus nội (Internal bus) là đường dẫn để truyền dữ liệu giữa các thanh ghi và ALU trong VXL
- Bus ngoại (External bus) dùng cho bên ngoài nối đến RAM, ROM và I/O
- Độ rộng của bus nội và ngoại có thể khác nhau.
- Thí dụ
  - 8088: bus nội là 16 bit, bus ngoại là 8 bit
  - 8086: bus nội là 16 bit, bus ngoại là 16 bit

2.1 SƠ ĐỒ KHỐI CPU 8 BIT CƠ BẢN


External input and output control lines
16-bit address bus
Memory address register
Hi
Lo
SP
PC
Accumulator A
Status reg
Reg B
Reg C
Reg D
Reg E
Reg H
Reg L
8-bit data bus
8-bit internal data bus
Control logic
Instruction register
Instruction decoder
Temp 1
Temp 2
In
In
ALU
Out

Sơ đồ chức năng và gán chân ở chip Z80



- Có 6 nhóm tín hiệu:
  - Bus địa chỉ 16 đường (A0 đến A15)
  - Bus dữ liệu 8 đường (D0 đến D7)
  - 6 đường điều khiển hệ thống

    $\overline{M1}$, $\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, $\overline{WR}$, $\overline{RFSH}$
  - 5 đường điều khiển CPU

    $\overline{HALT}$, $\overline{WAIT}$, $\overline{INT}$, $\overline{NMI}$, $\overline{RESET}$
  - 2 đường điều khiển bus CPU ($\overline{BUSREQ}$, $\overline{BUSACK}$)
  - 3 đường dành cho nguồn cấp điện và xung nhịp (+5V, GND, và CLK)

Mô tả chân Z80

A15-A0 :

Bus địa chỉ (xuất, tích cực cao, 3-state).
Dùng để truy cập bộ nhớ và các cổng I/O
Trong chu kỳ làm tươi I được đặt trên bus này.

D7-D0 :

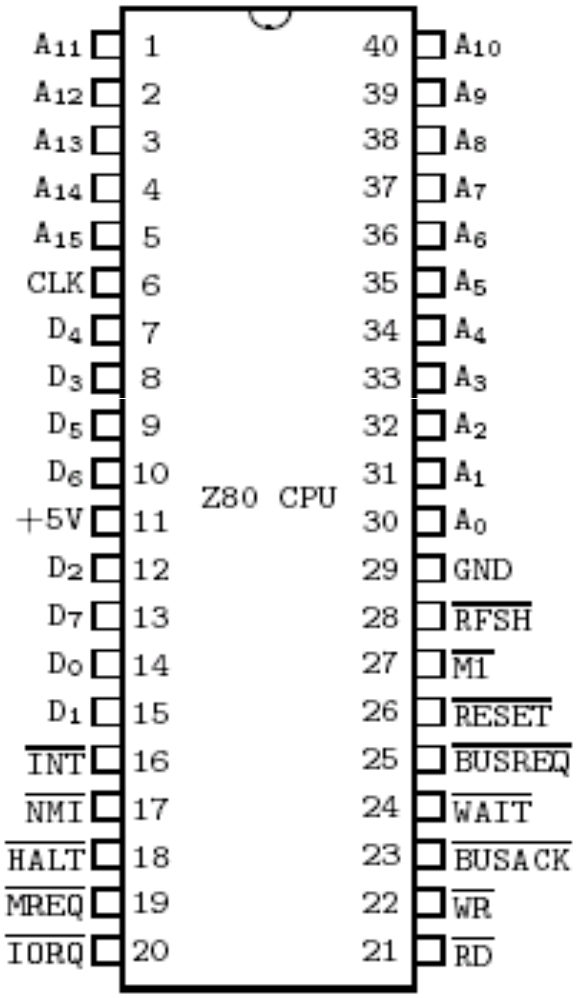
Bus dữ liệu (nhập/xuất, tích cực cao, 3-state). Dùng để trao đổi dữ liệu với bộ nhớ , I/O và ngắt.

RD:

Đọc (xuất, tích cực thấp, 3-state) cho biết CPU muốn đọc dữ liệu từ bộ nhớ hay I/O

WR:

Ghi (xuất, tích cực thấp, 3-state) cho biết bus dữ liệu CPU giữ dữ liệu hợp lệ sẽ được cất vào bộ nhớ hay thiết bị I/O.



Mô tả chân Z80

MREQ

Memory Request (output, active Low, 3-state).
Indicates memory read/write operation. See M1

IORQ

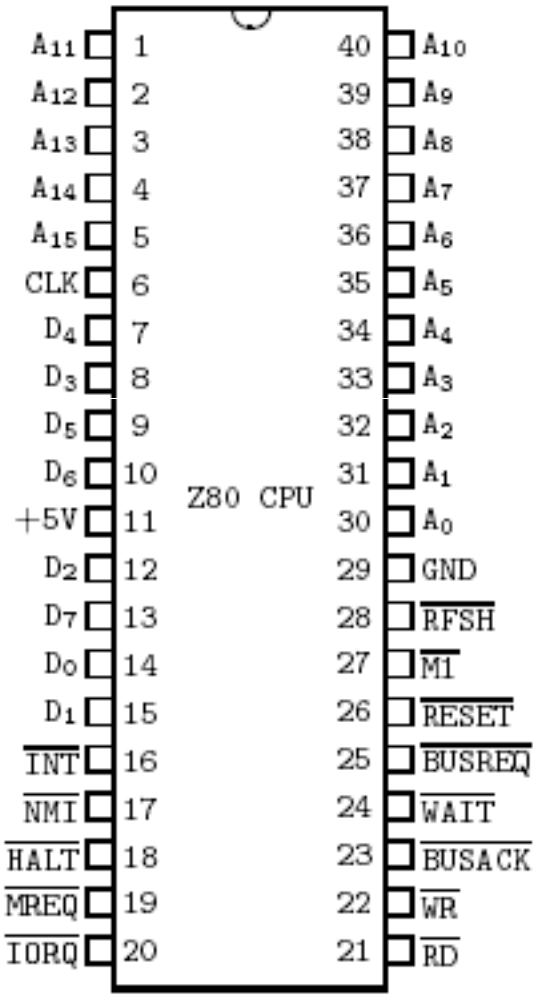
Input/Output Request(output,active Low,3-state)
Indicates I/O read/write operation. See M1

M1

Machine Cycle One (output, active Low).
Together with MREQ indicates opcode fetch cycle
Together with IORQ indicates an Int Ack cycle

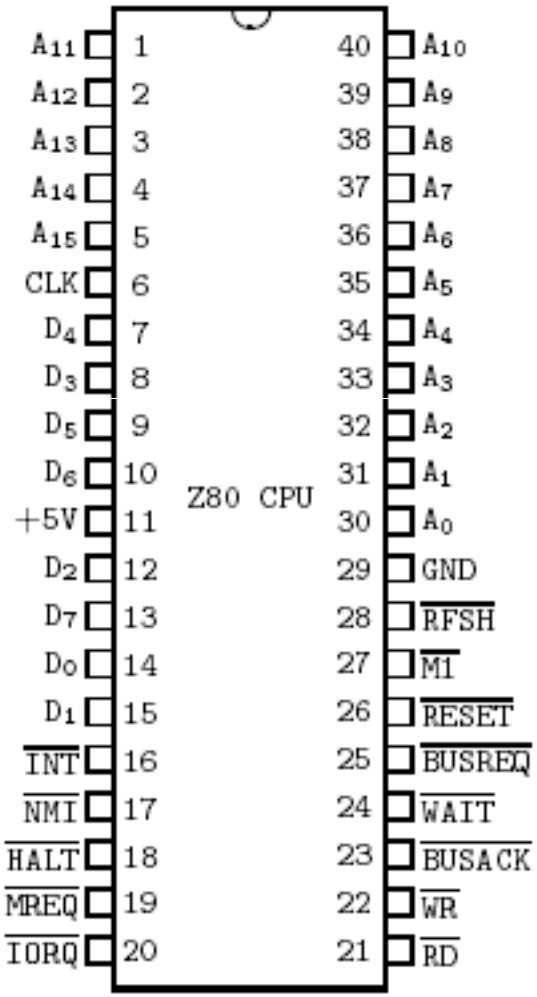
RFSH

Refresh (output, active Low).
Together with MREQ indicates refresh cycle.
Lower 7-bits address is refresh address to DRAM



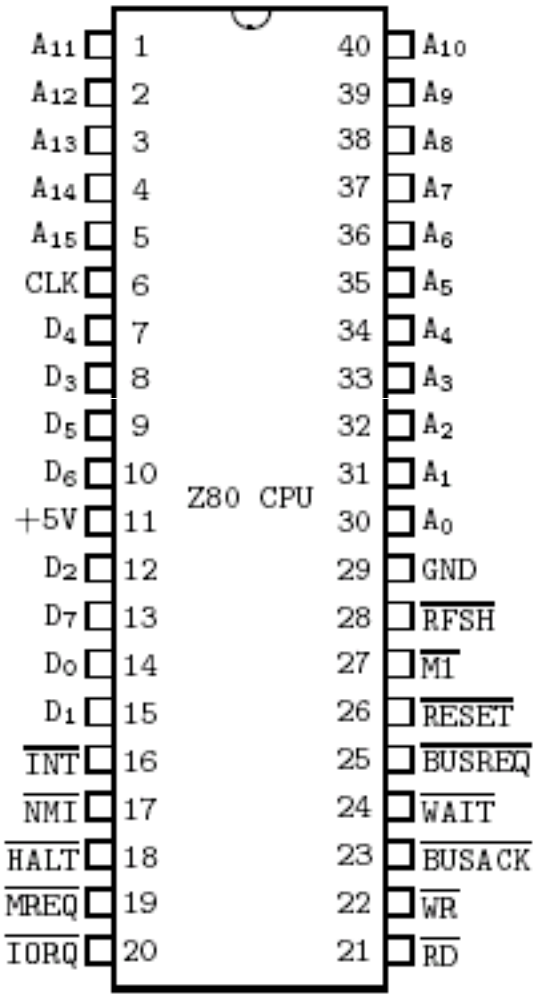
Mô tả chân Z80

- **INT**
  - Interrupt Request (input, active Low).
  - Interrupt Request is generated by I/O devices.
  - Checked at the end of the current instruction
  - If flip-flop (IFF) is enabled.
- **NMI**
  - Non-Maskable Interrupt
  - (Input, negative edge-triggered).
  - Higher priority than INT.
  - Recognized at the end of the current Instruction
  - Independent of the status of IFF
  - Forces the CPU to restart at location 0066H.



Mô tả chân Z80

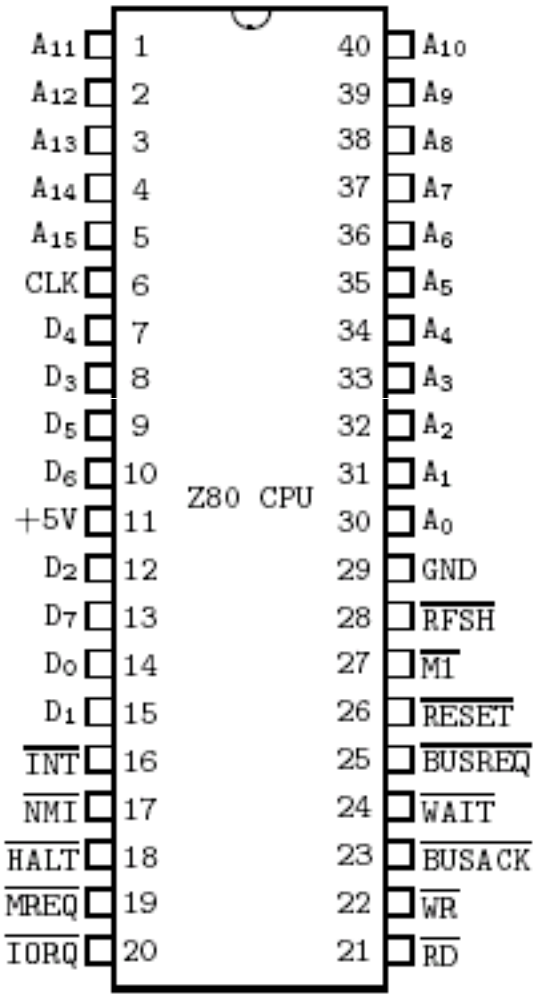
- BUSREQ
  - Bus Request (input, active Low).
  - higher priority than NMI
  - recognized at the end of the current machine cycle.
  - forces the CPU address bus, data bus, and MREQ, IORQ, RD, and WR to high-imp.
- BUSACK
  - Bus Acknowledge (output, active,Low)
  - indicates to the requesting device that address, data, and control signals MREQ, IORQ, RD, and WR have entered their high-impedance states.



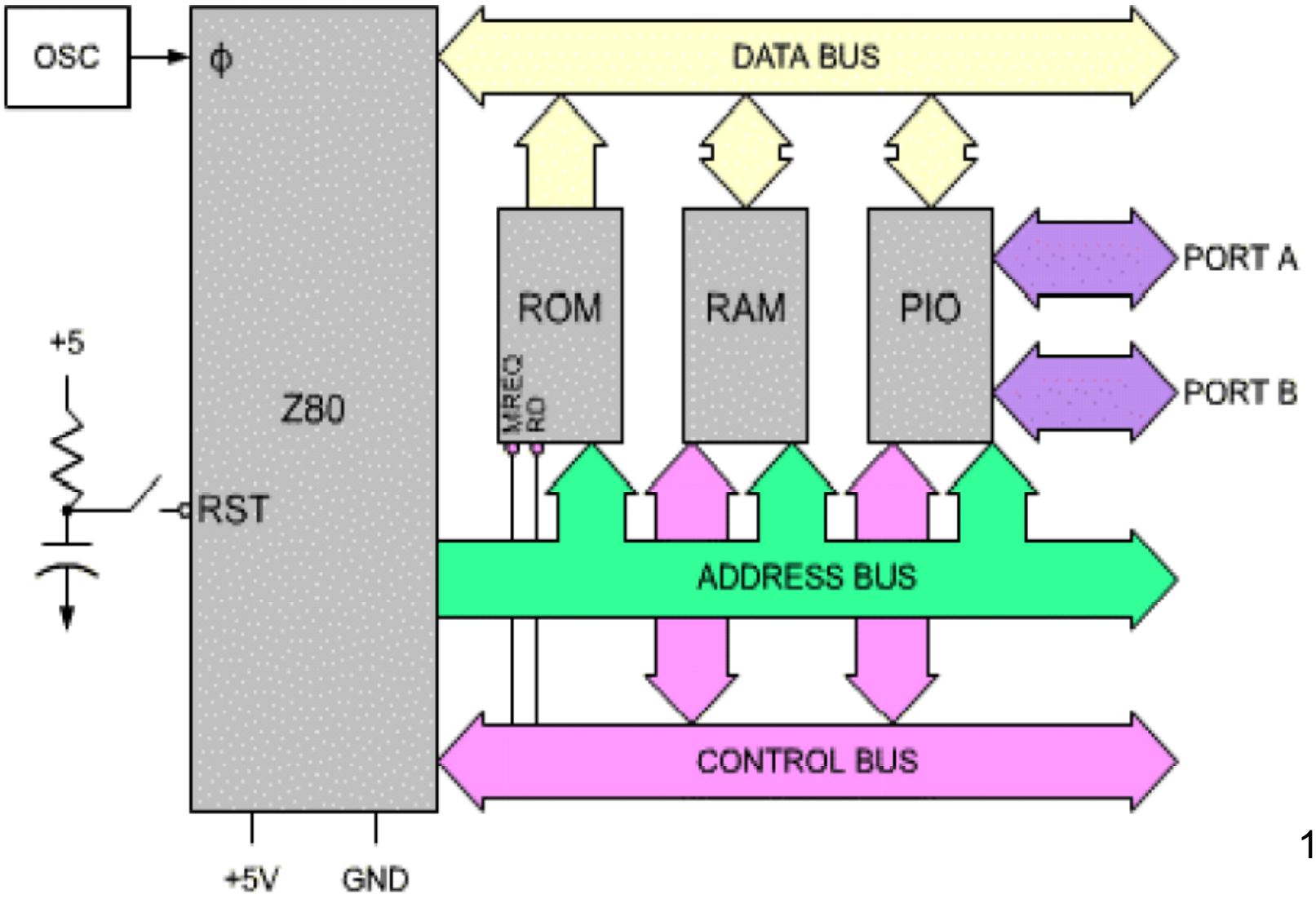
Mô tả chân Z80

RESET

Reset (input, active Low).
RESET initializes the CPU as follows:
Resets the IFF
Clears the PC and registers I and R
Sets the interrupt status to Mode 0. During reset time, the address and data bus go to a high-impedance state And all control output signals go to the inactive
state.
must be active for a minimum of three full clock cycles before the reset operation is complete.

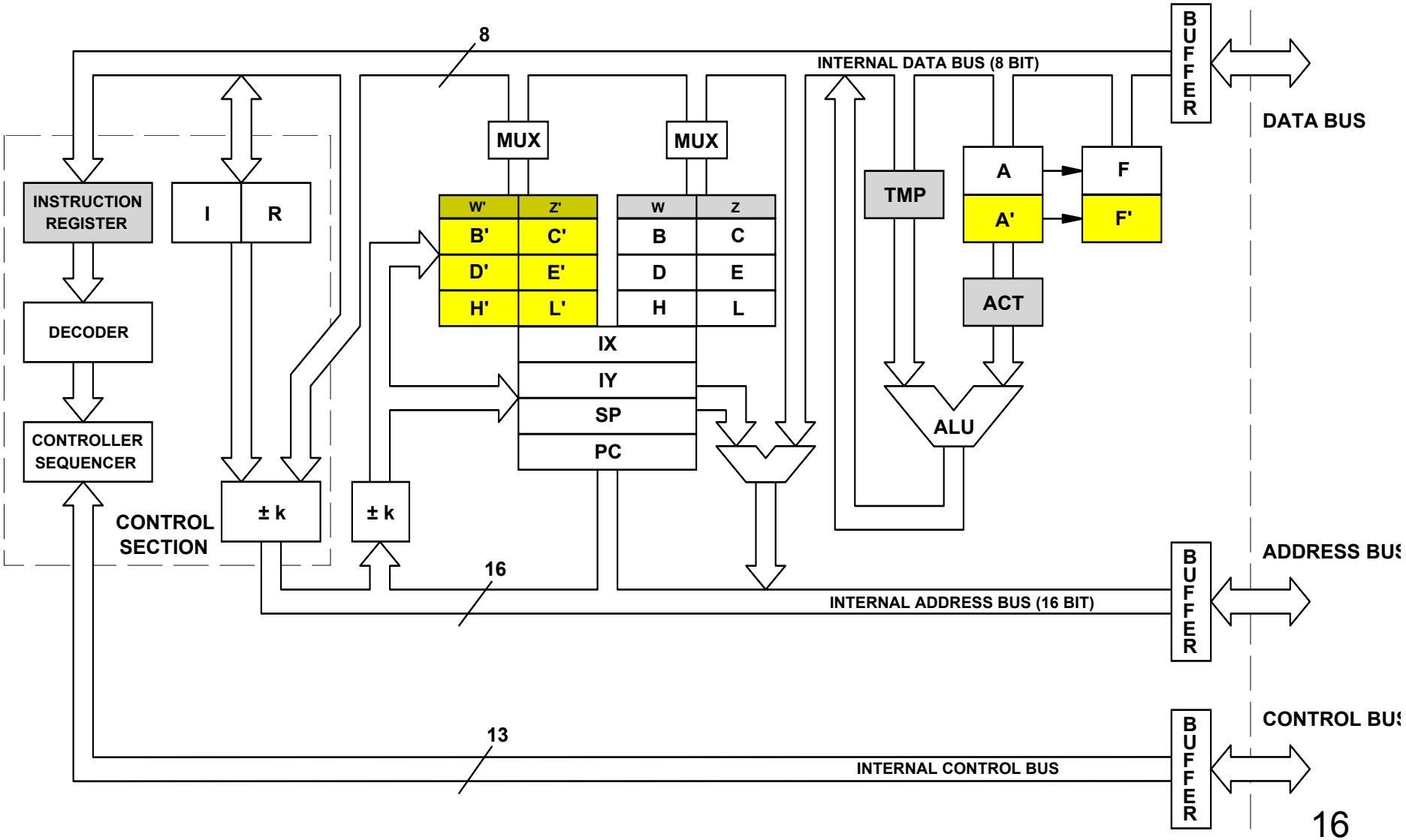


Kiến trúc hệ thống

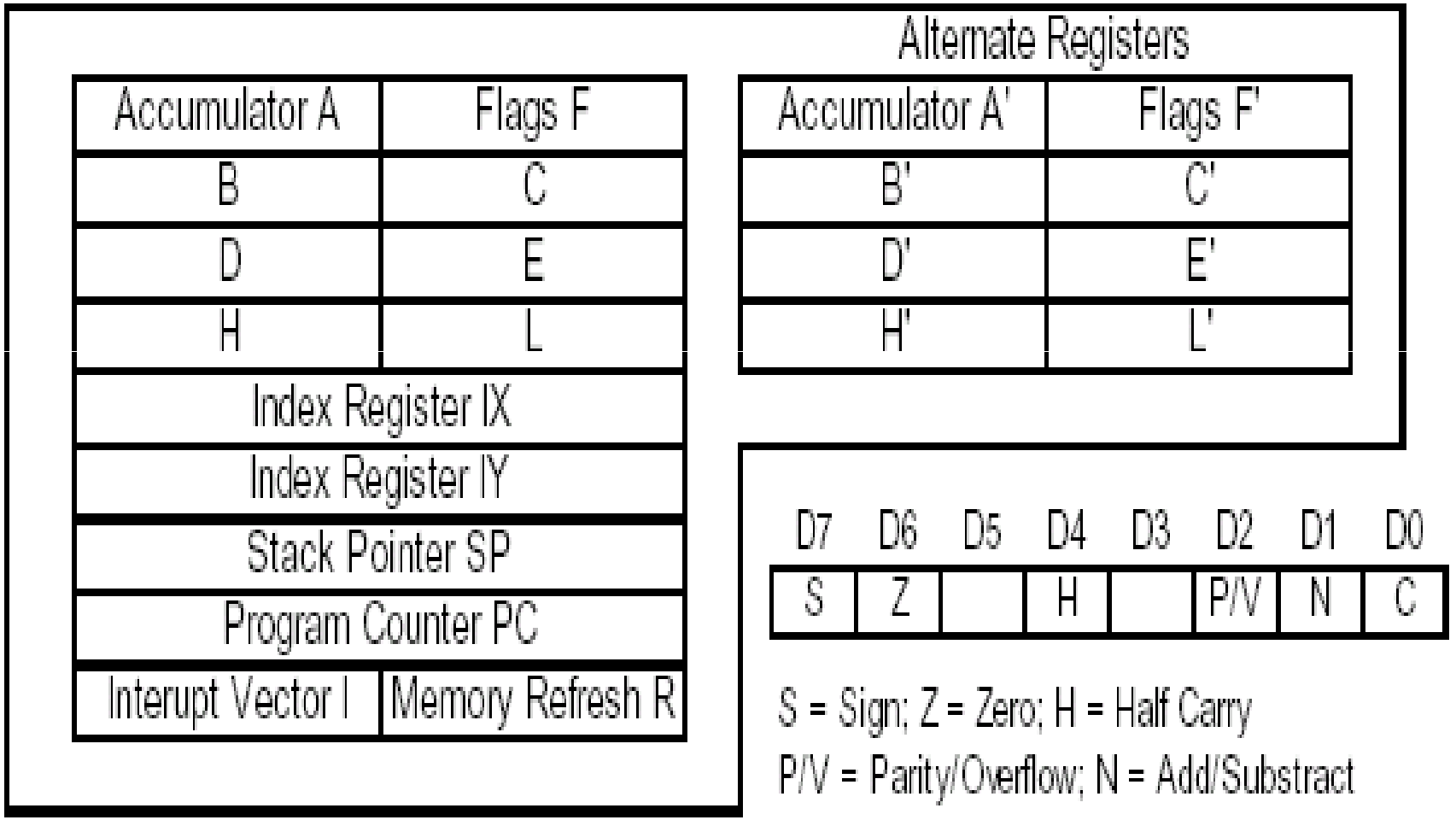


2.2 TỔ CHỨC CÁC THANH GHI

Z80 CPU


8
INTERNAL DATA BUS (8 BIT)
BUFFER
DATA BUS
MUX
MUX
INSTRUCTION REGISTER
I
R
W'
Z'
B'
C'
D'
E'
H'
L'
W
Z
B
C
D
E
H
L
TMP
A
A'
F
F'
ACT
IX
IY
SP
PC
ALU
DECODER
CONTROLLER SEQUENCER
CONTROL SECTION
± k
± k
16
INTERNAL ADDRESS BUS (16 BIT)
BUFFER
ADDRESS BUS
13
INTERNAL CONTROL BUS
BUFFER
CONTROL BUS

Mô hình lập trình Z80



Tập thanh ghi

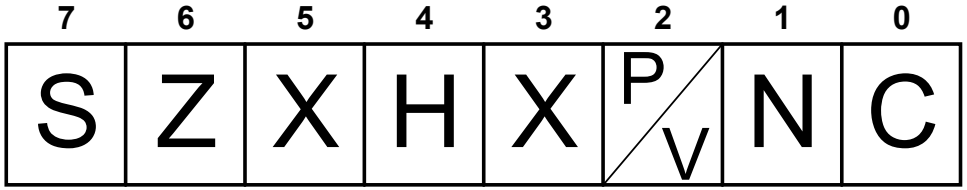
- A : Accumulator Register
- F : Flag register
- Two sets of six general-purpose registers
  - may be used individually as 8-bit A F B C D E H L (A' F' B' C' D' E' H' L')
  - or in pairs as 16-bit registers AF BC DE HL (AF' BC' DE' HL')
- The Alternative registers (A' F' B' C' D' E' H' L') not visible to the programmer but can access via:
  - EXX (BC)<->(BC') , (DE)<->(DE') , (HL)<->(HL')
  - EX AF, AF' (AF)<->(AF')

  what is this instruction useful for?

Tập thanh ghi (tt)

- 4 16-bit registers hold memory address (pointers)
  - index registers (IX) and (IY) are 16-bit memory pointers
  - 16 bit stack pointer (SP)
  - Program counter (PC)
- Program counter (PC)
  - PC points to the next opcode to be fetched from ROM
  - when the µP places an address on the address bus to fetch the byte from memory, it then increments the program counter by one to the next location
- Special purpose registers
  - I : Interrupt vector register.
  - R : memory Refresh register

Thanh ghi cờ (Flag Register)



S Sign Flag (1:negativ)*

Z Zero Flag (1:Zero)

H Half Carry Flag (1: Carry from Bit 3 to Bit 4)**

P Parity Flag (1: Even)

V Overflow Flag (1:Overflow)*

N Operation Flag (1:previous Operation was subtraction)**

C Carry Flag (1: Carry from Bit n-1 to Bit n,
with n length of operand)

*: 2-complement number representation

: **used in DAA-operation for BCD-arithmetic

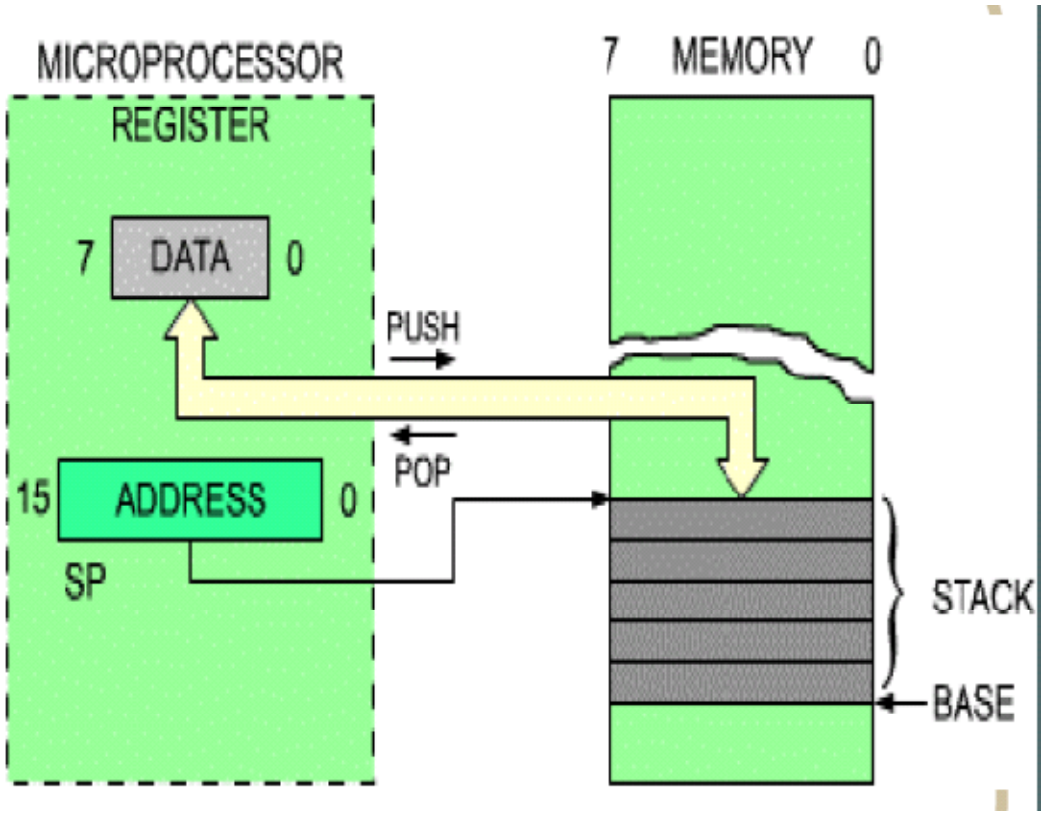
DAA - Decimal Adjust Accumulator

Adjusts the content of the Accumulator A for BCD addition and subtraction operations such as ADD, ADC, SUB, SBC, and NEG according to the table:

	before DAA					after DAA	
Op	N	C	Bits 4-7	H	Bits 0-3	A=A+..	C
ADD ADC	0	0	0-9	0	0-9	00	0
	0	0	0-8	0	A-F	06	0
	0	0	0-9	1	0-3	06	0
	0	0	A-F	0	0-9	60	1
	0	0	9-F	0	A-F	66	1
	0	0	A-F	1	0-3	66	1
	0	1	0-2	0	0-9	60	1
	0	1	0-2	0	A-F	66	1
	0	1	0-3	1	0-3	66	1
SUB SBC NEG	1	0	0-9	0	0-9	00	0
	1	0	0-8	1	6-F	FA	0
	1	1	7-F	0	0-9	A0	1
	1	1	6-F	1	6-F	9A	1

Stack Pointer (SP)

- Dùng làm con trỏ chỉ đến stack bộ nhớ ngoài
- Khi đưa dữ liệu vào stack (PUSH), SP giảm đi 1.
- Khi lấy dữ liệu ra khỏi stack (POP), SP tăng thêm 1.

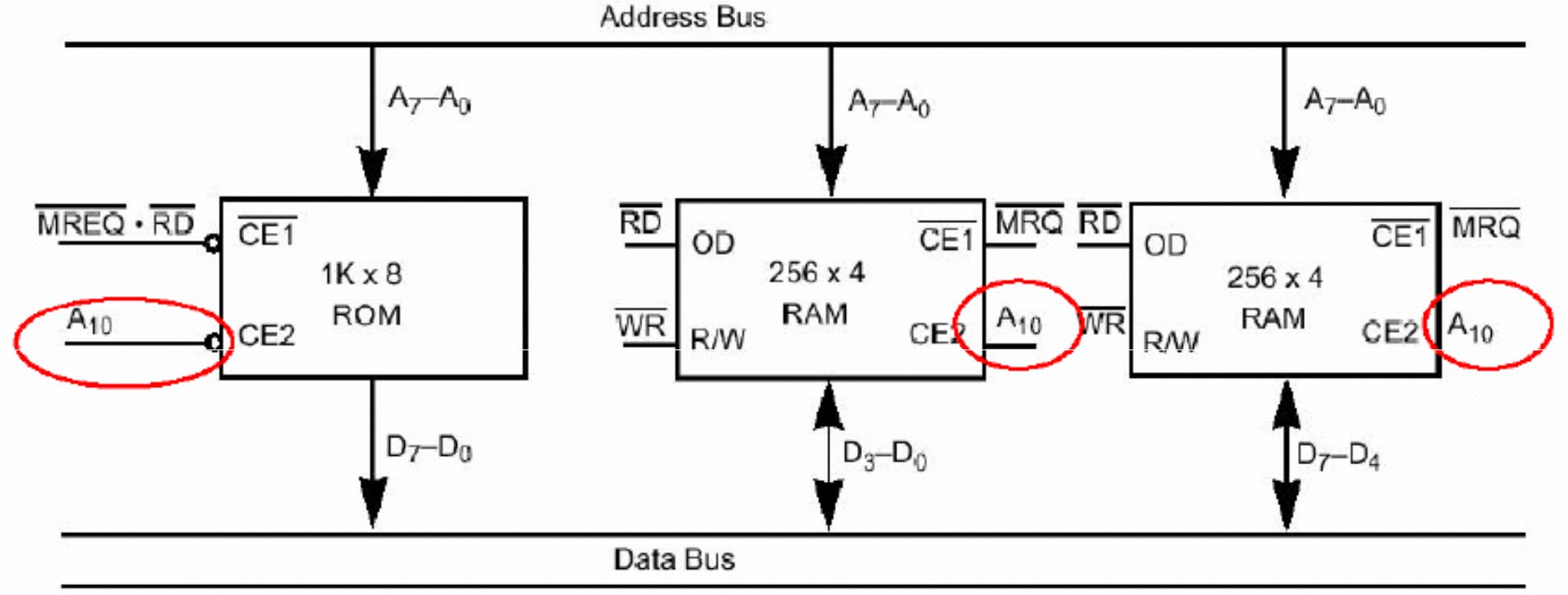


2.3 TỔ CHỨC BỘ NHỚ

Tổ chức bộ nhớ

- Tổ chức bộ nhớ tùy theo ứng dụng khác nhau ta có các tổ chức khác nhau và tùy theo ROM, SRAM, DRAM mà ta có các kết nối tín hiệu điều khiển khác nhau.
- Với địa chỉ 16 đường (A0 đến A15), Z80 có thể làm việc đến tối đa 64KB bộ nhớ.
- Tám đường địa chỉ thấp (A0 đến A7) cũng được dùng để truy cập tới 256 cổng I/O.
- Để minh họa phần này ta sẽ khảo sát một số thí dụ.

Giao tiếp ROM 1KB và 2 RAM 256 x 4

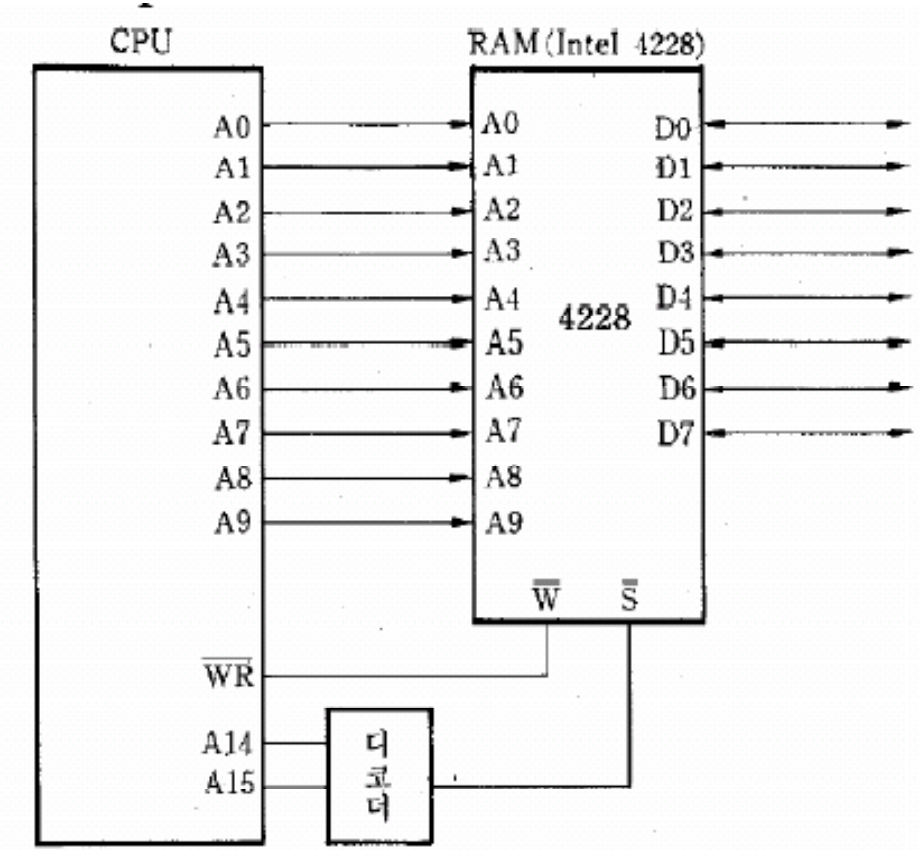


Ta muốn Z80 kết nối với các bộ nhớ (với các chip ROM 1K x 8 và RAM 256 x 4) theo bảng bộ nhớ sau

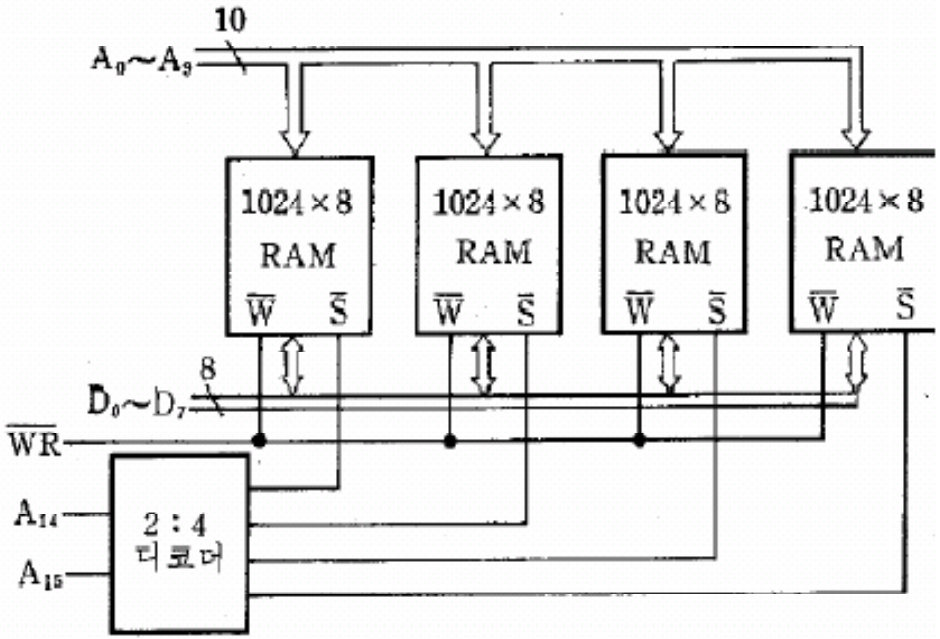
ROM 1 KB : 0000H–03FFH

RAM 256 bytes : 0400H–04FFH

Giao tiếp với RAM

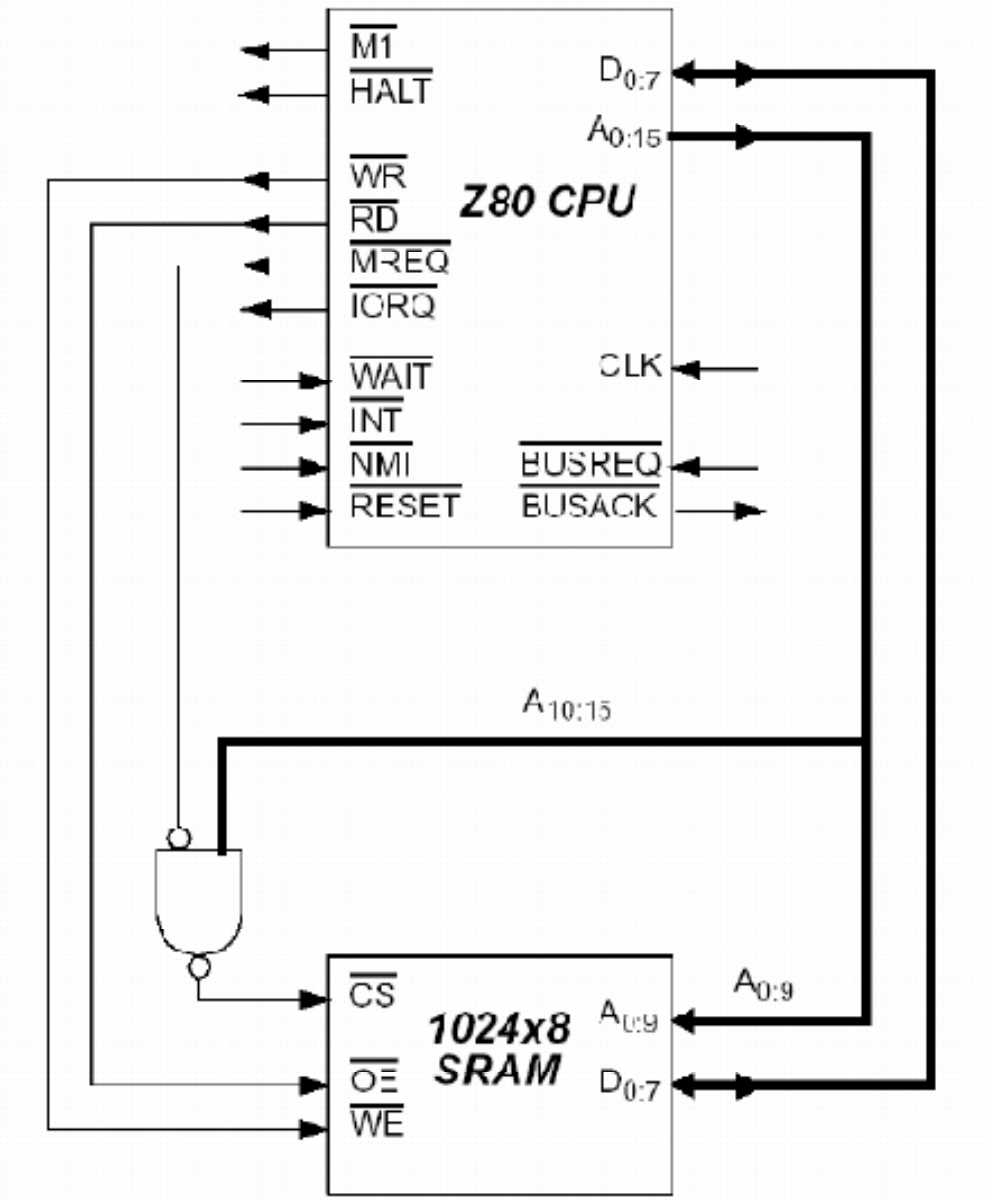


(a) Với RAM 1KB

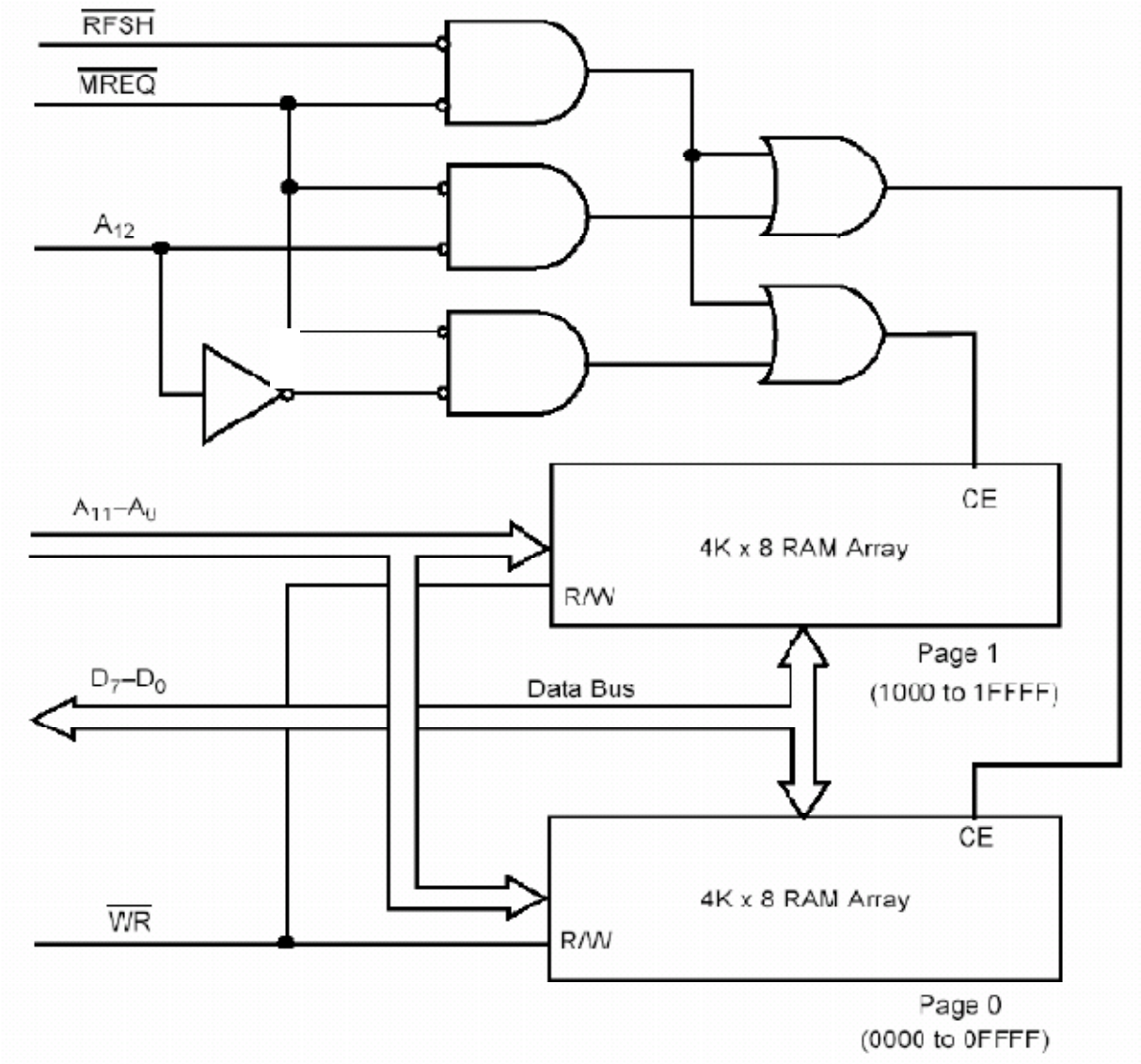


(b) Với RAM 4 KB

Giao tiếp với SRAM 1KB

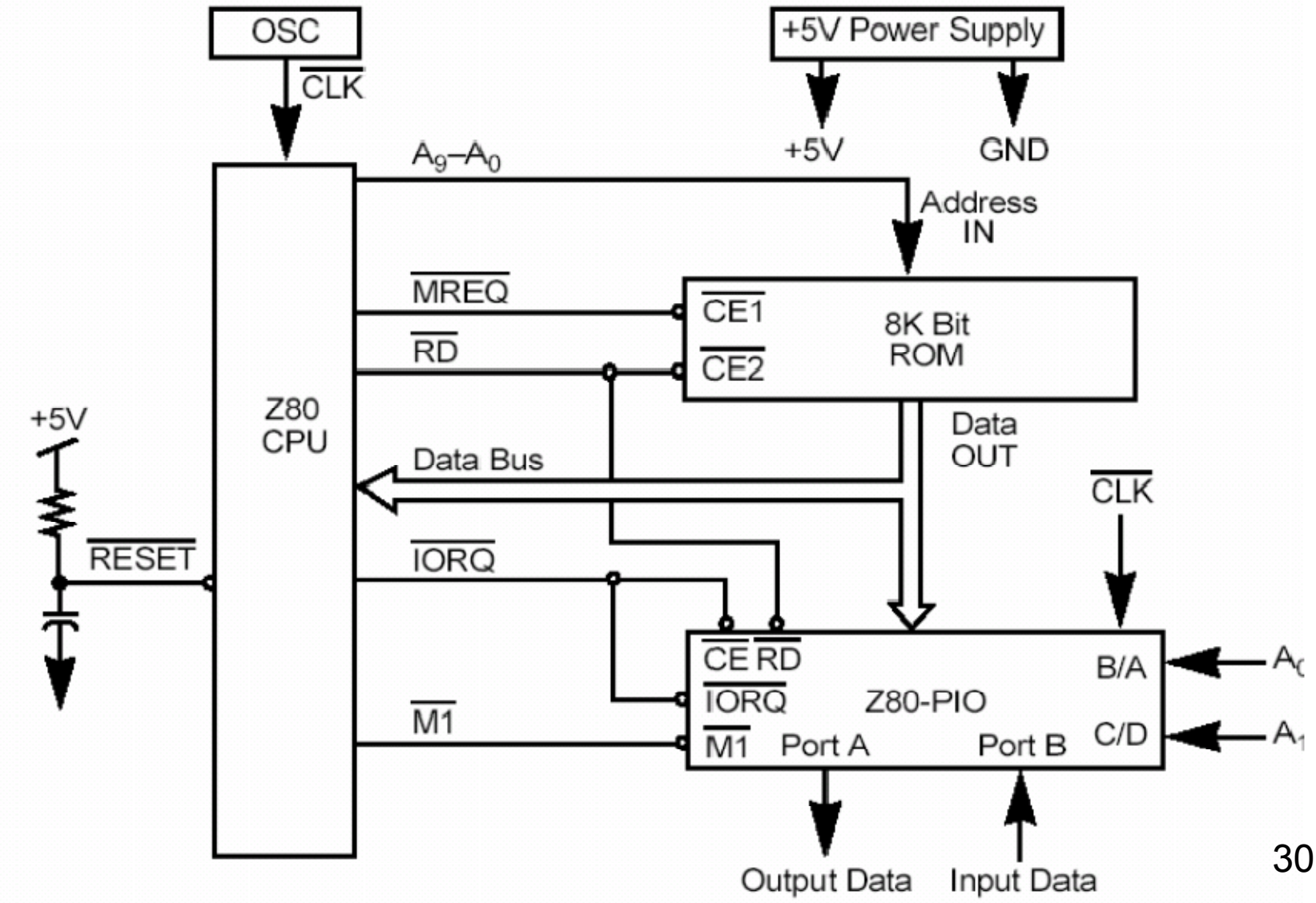


Thí dụ giao tiếp DRAM 8 KB xây dựng từ các DRAM 4KB



2.4 GHÉP NỐI BUS HỆ THỐNG

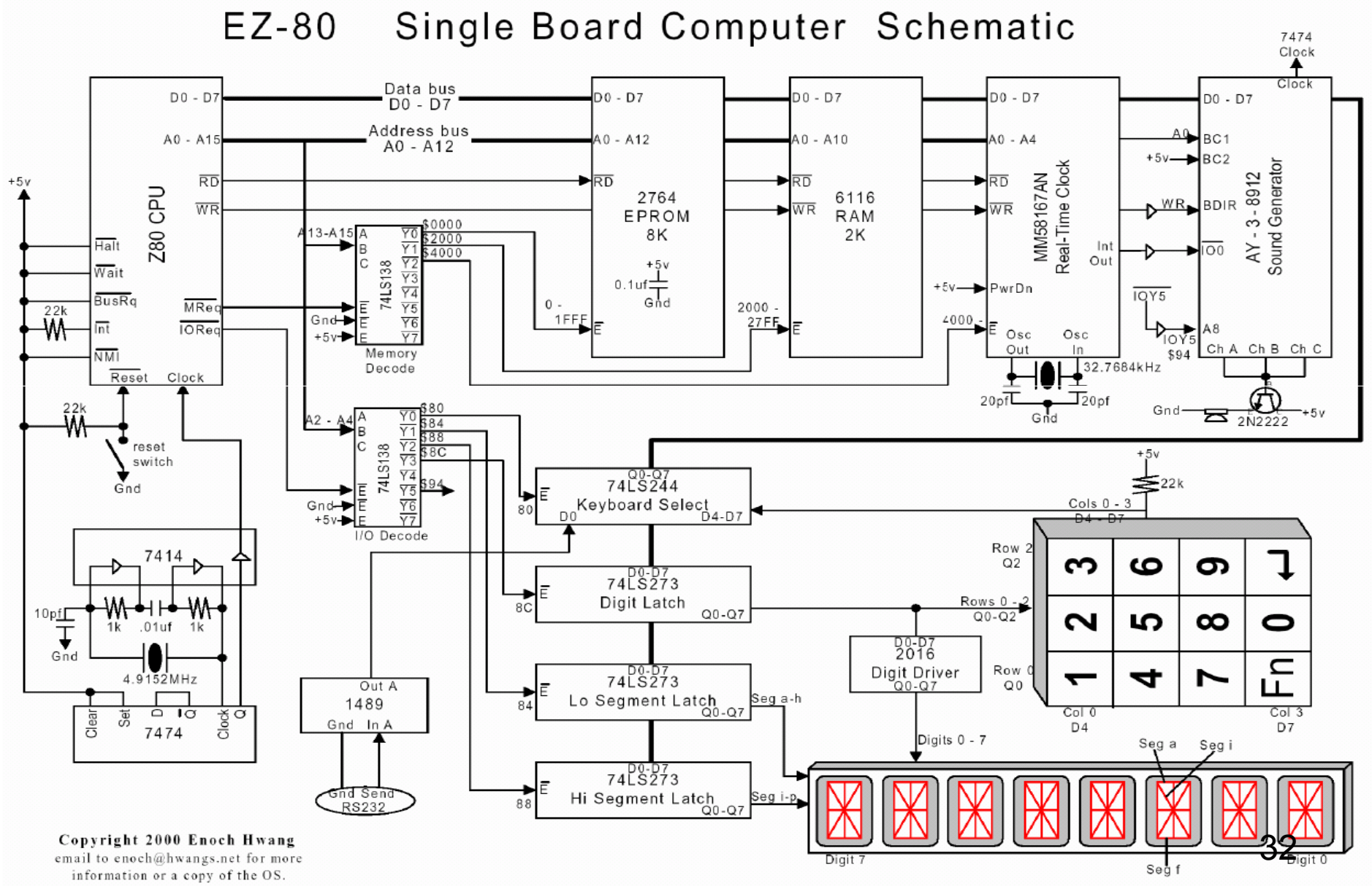
Hệ máy tính Z80 tối thiểu



Mở rộng giao tiếp cho Z80

- Để có thể mở rộng giao tiếp cho Z80, hãng Zilog đã phát triển các chip hỗ trợ sau:
  - **Z80 PIO** là bộ điều khiển I/O song song, nó làm cho Z80 mở rộng thêm thành 2 cổng I/O song song 8 bit. Chip còn có thêm đường tạo ngắt cho Z80 và cho phép nối logic OR các chân này lại.
  - **Z80 CTC** là mạch bộ đếm-định thì (counter-timer circuit) để cho người thiết kế hệ thống Z80 sử dụng nó thực hiện các chức năng đếm và định thì.
  - **Z80 SIO** là mạch nhập/xuất nối tiếp (Serial Input/Output Circuit), chip này cung cấp cho hệ Z80 với 2 cổng nối tiếp mà có thể sử dụng để liên lạc với các thiết bị ngoại vi nối tiếp khác.
  - **Z80 DMA** thực hiện việc truy cập bộ nhớ trực tiếp với thiết bị ngoài.

Thí dụ sơ đồ phần cứng một kit dựa trên Z80



2.5 CHU KỲ BUS, CHU KỲ MÁY

Chu kỳ lệnh, chu kỳ máy và các trạng thái T

- Chu kỳ lệnh là thời gian cần để hoàn tất việc thực thi một lệnh.

- Chu kỳ máy được định nghĩa là thời gian cần hoàn tất một tác vụ truy cập bộ nhớ, truy cập I/O,…

  (Với Z80, chu kỳ máy có thể kéo dài từ 3 đến 6 chu kỳ xung nhịp)

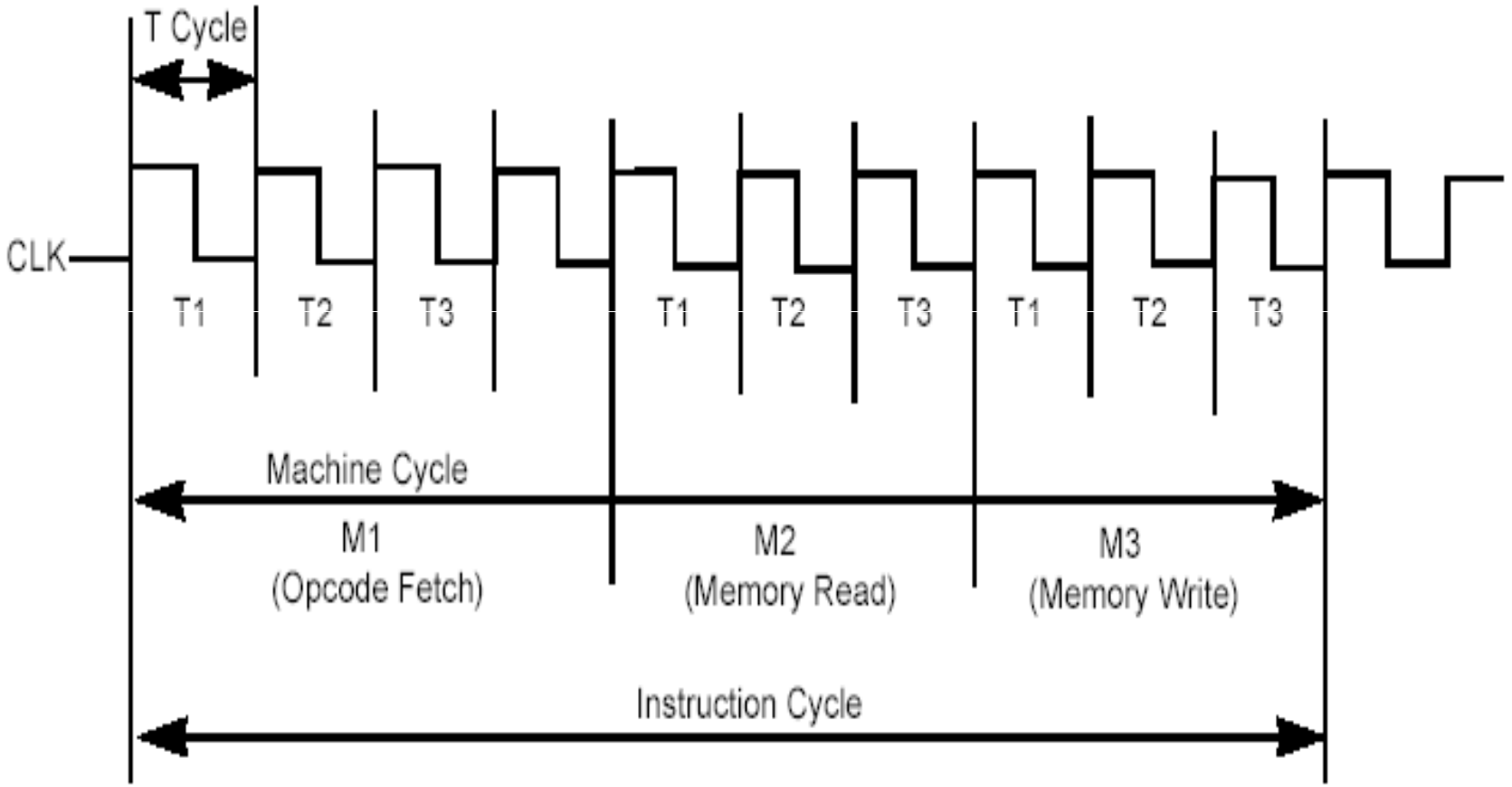
- T-state = 1/f (f: tần số Clock của Z80)
  - f= 4MHZ → T-state=0.25 uS

chu kỳ máy

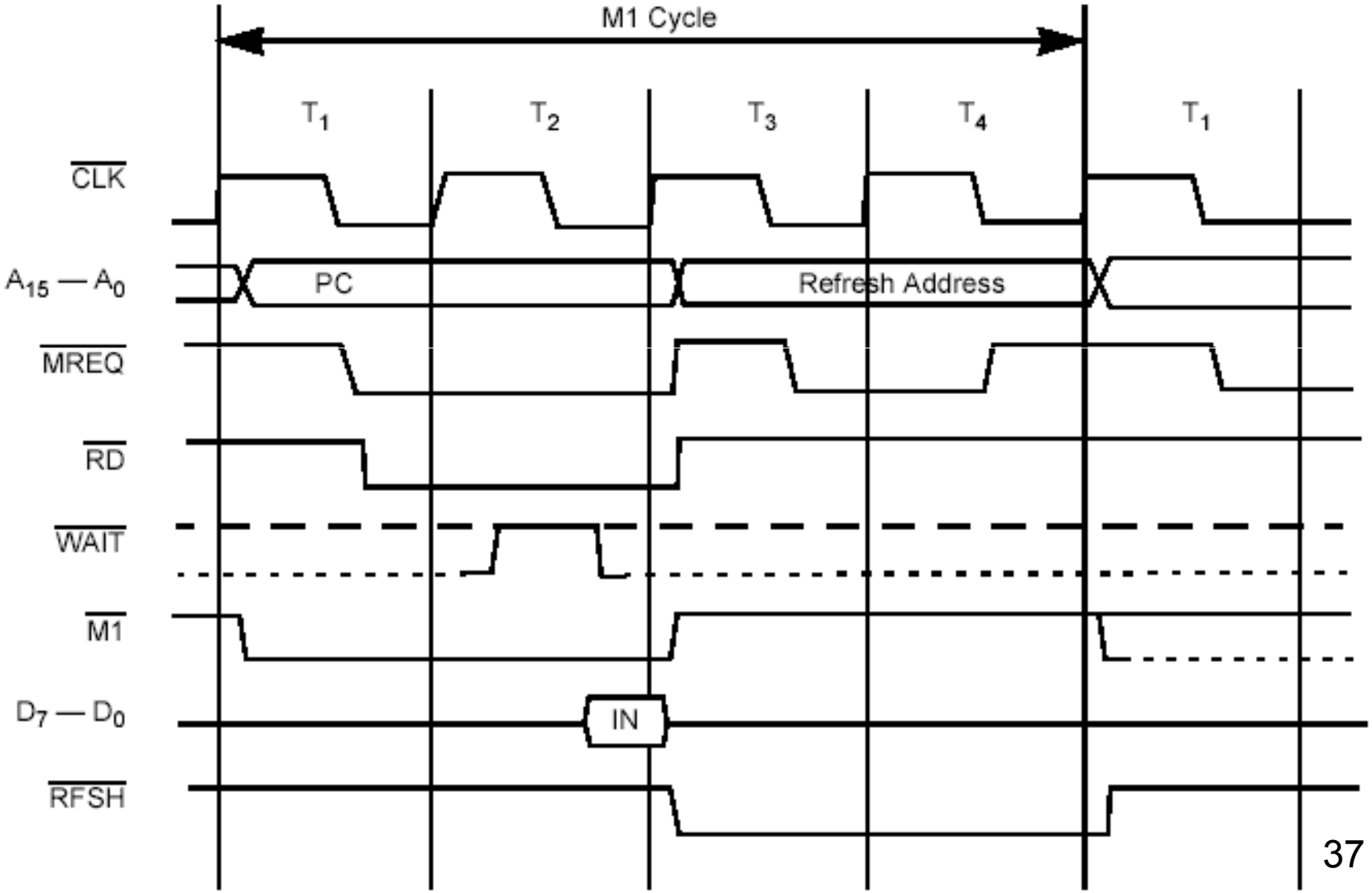
Có 7 chu kỳ máy với Z80:

1. Nhận mã lệnh (chu kỳ M1)
2. Đọc hoặc ghi dữ liệu bộ nhớ
3. Đọc hoặc ghi I/O
4. Yêu cầu/ghi nhận bus (Bus Request/Acknowledge)
5. Yêu cầu/ghi nhận NMI
6. Thoát khỏi lệnh HALT

Thí dụ định thì CPU Z80



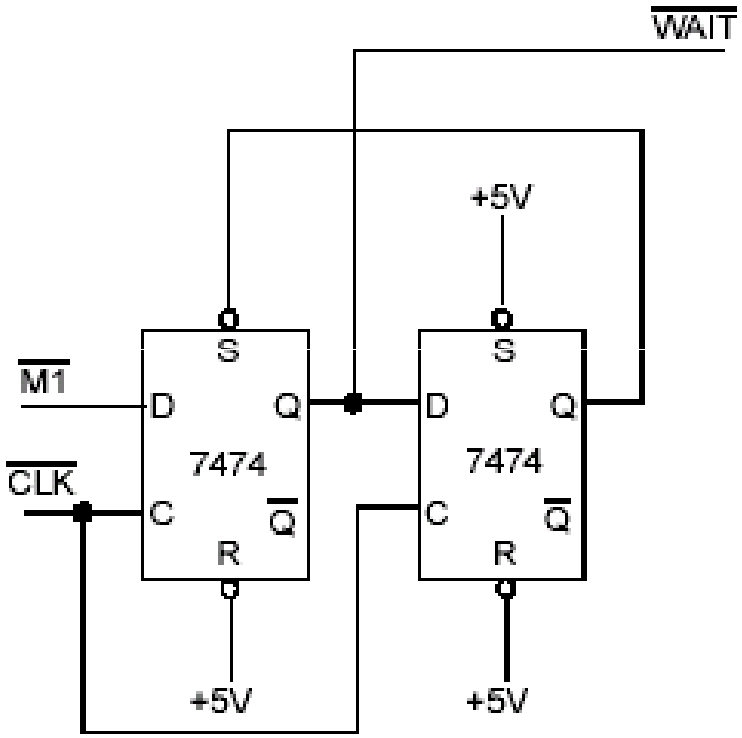
Chu kỳ nhận lệnh (chu kỳ M1)

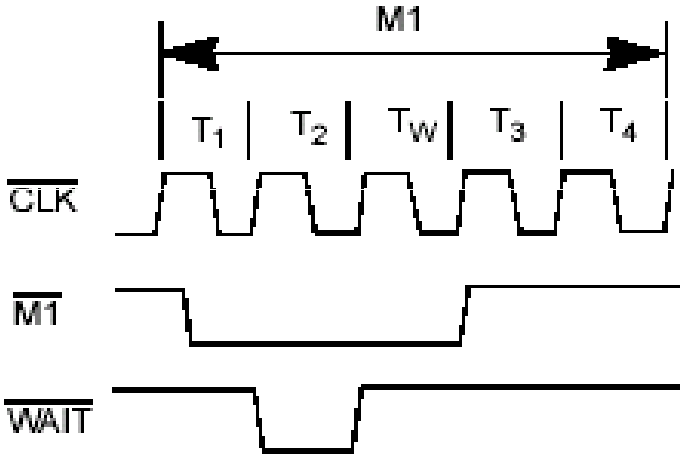


Thanh ghi R (Refresh)

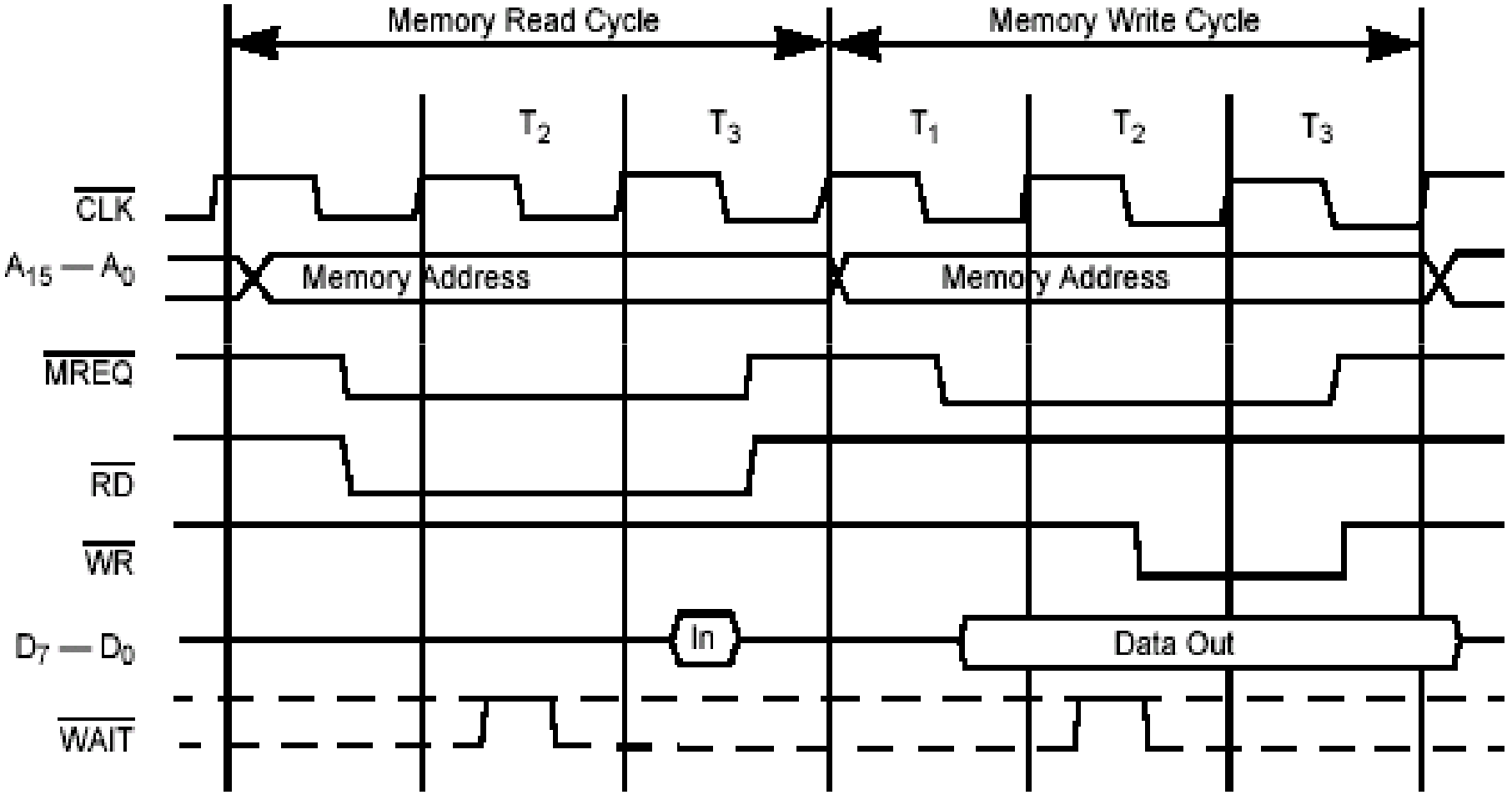
- Được tăng ở mọi chu kỳ M1.
- Bit 7 của nó không bao giờ bị thay đổi bởi M1; chỉ có 7 bit thấp tham gia trong phép cộng. Vì vậy bit 7 giữ nguyên trị cũ.
- Ta chỉ có thể thay đổi bit 7 bằng lệnh LD R,A
- LD A,R và LD R,A truy cập thanh ghi R sau khi nó được tăng
- R thường được dùng làm giá trị ngẫu nhiên trong chương trình nhưng dĩ nhiên nó thật sự không ngẫu nhiên.

Thêm một trạng thái đợi vào chu kỳ T1

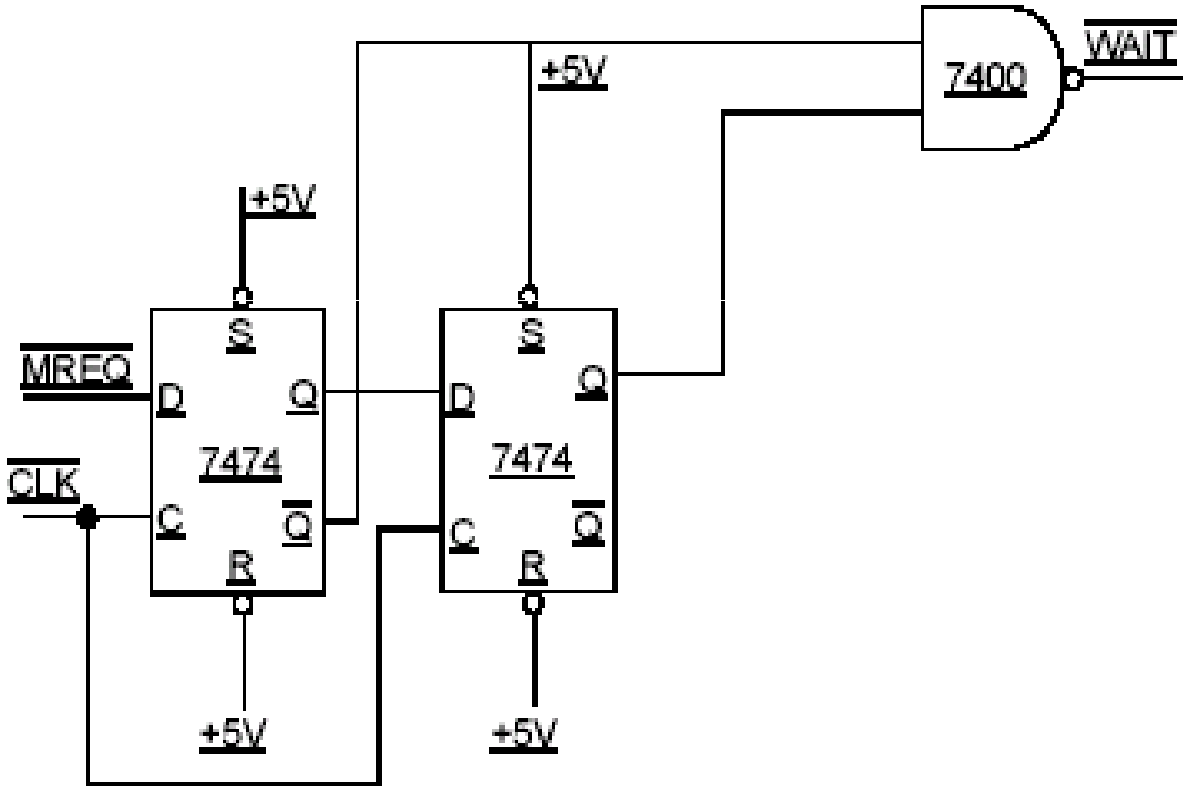


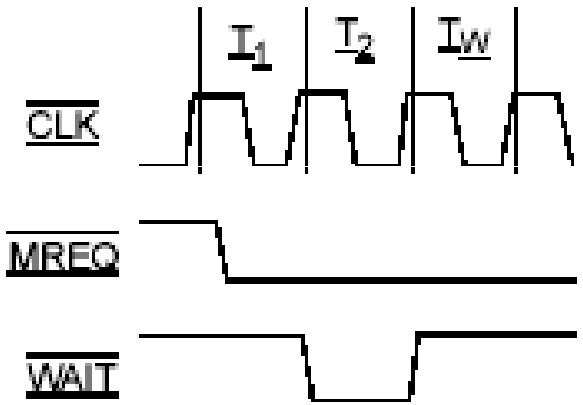


Chu kỳ đọc hoặc ghi bộ nhớ

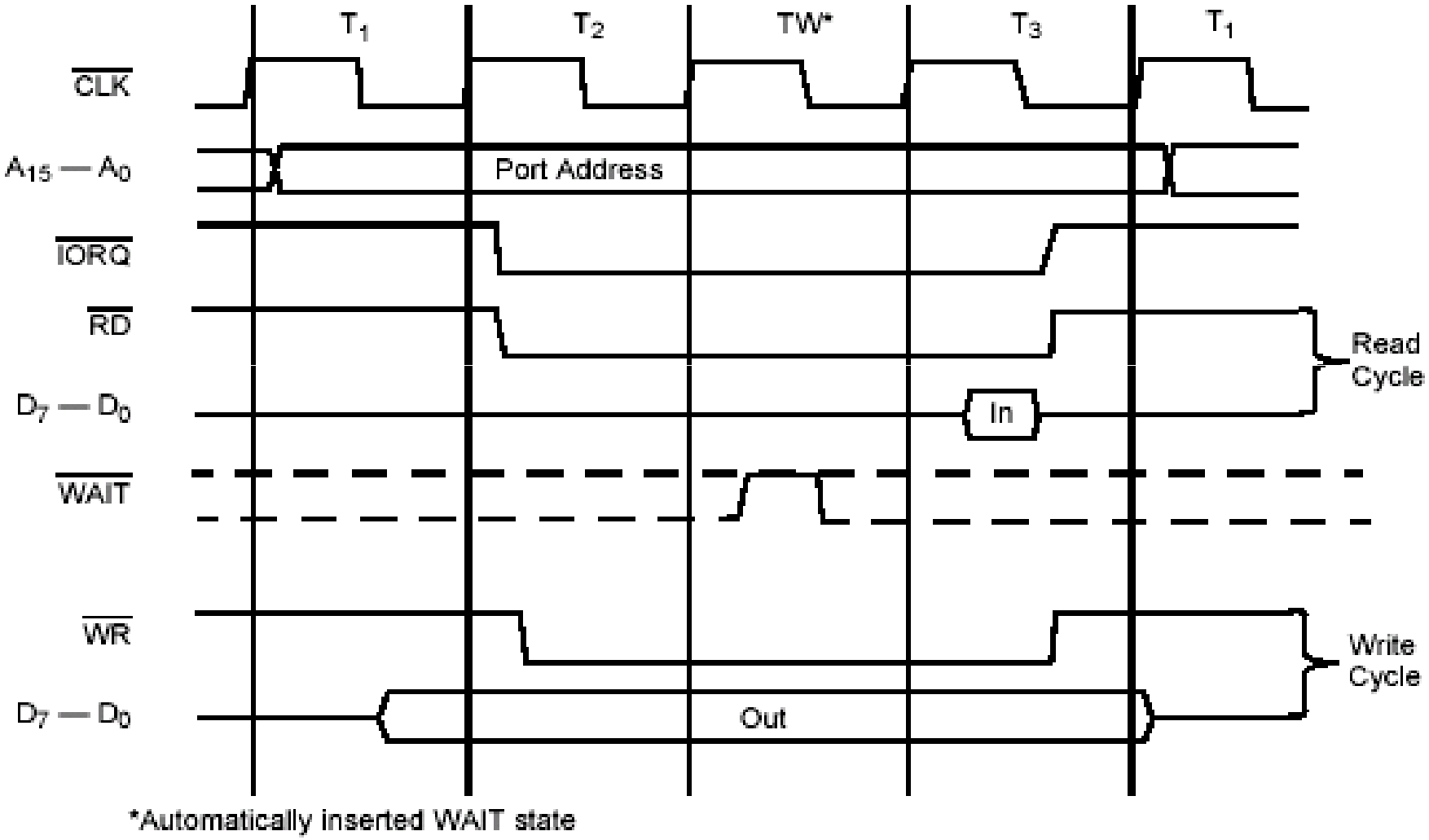


Thêm trạng thái đợi vào bất kỳ chu kỳ bộ nhớ nào



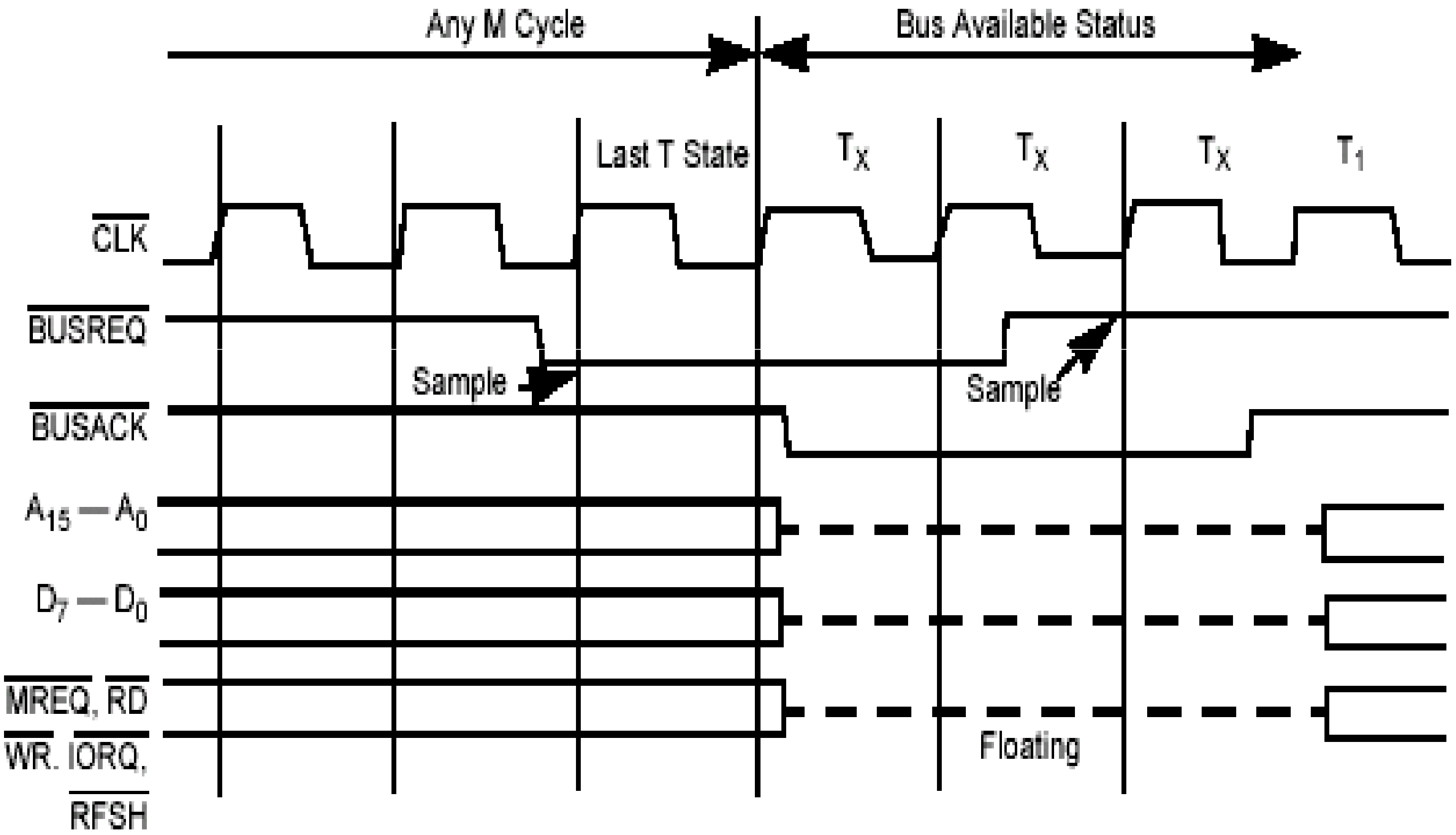


Chu kỳ nhập hoặc xuất (Input or Ouput Cycle)

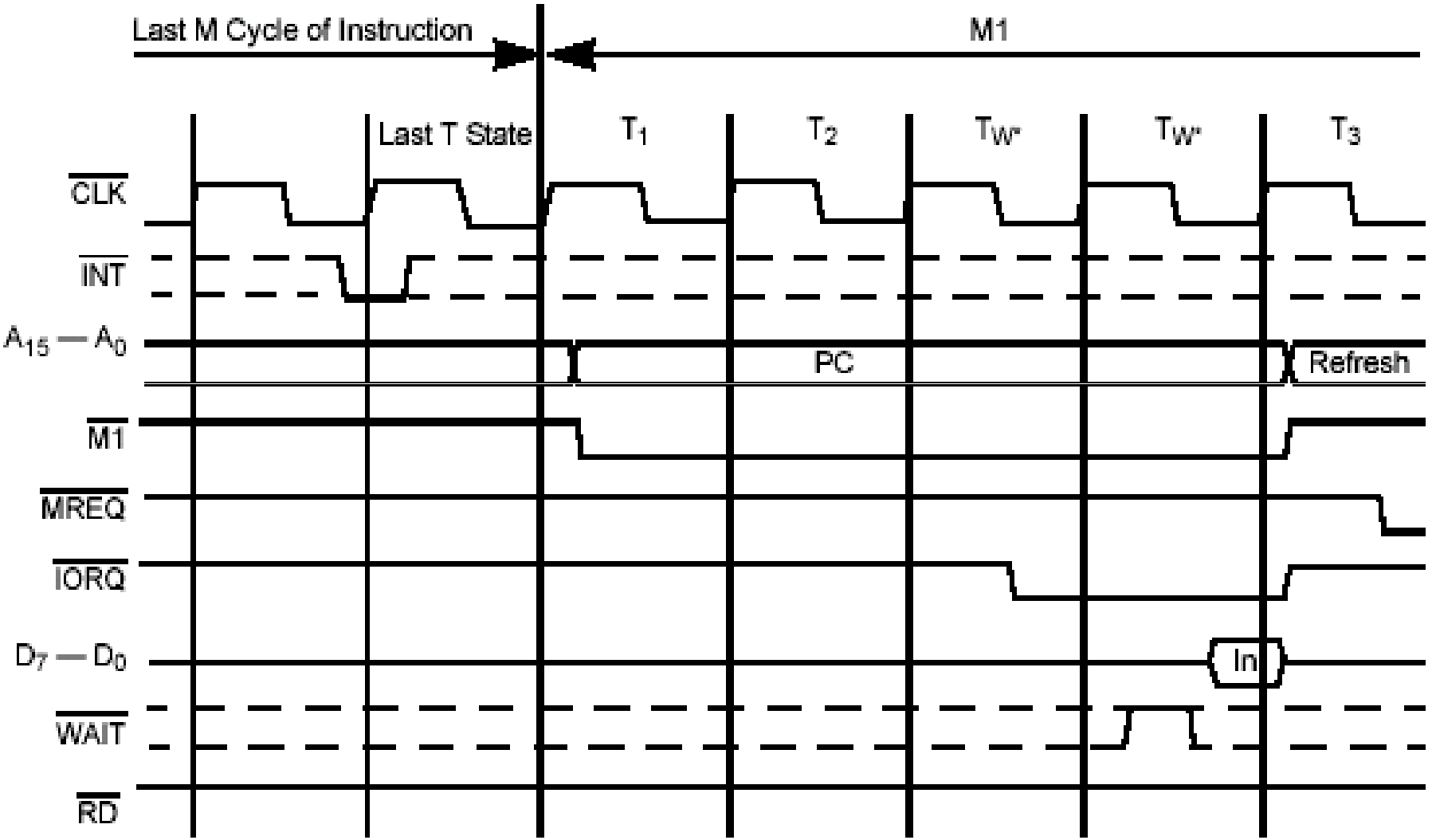


During I/O operations a single wait state is automatically inserted

Chu kỳ yêu cầu bus/ ghi nhận

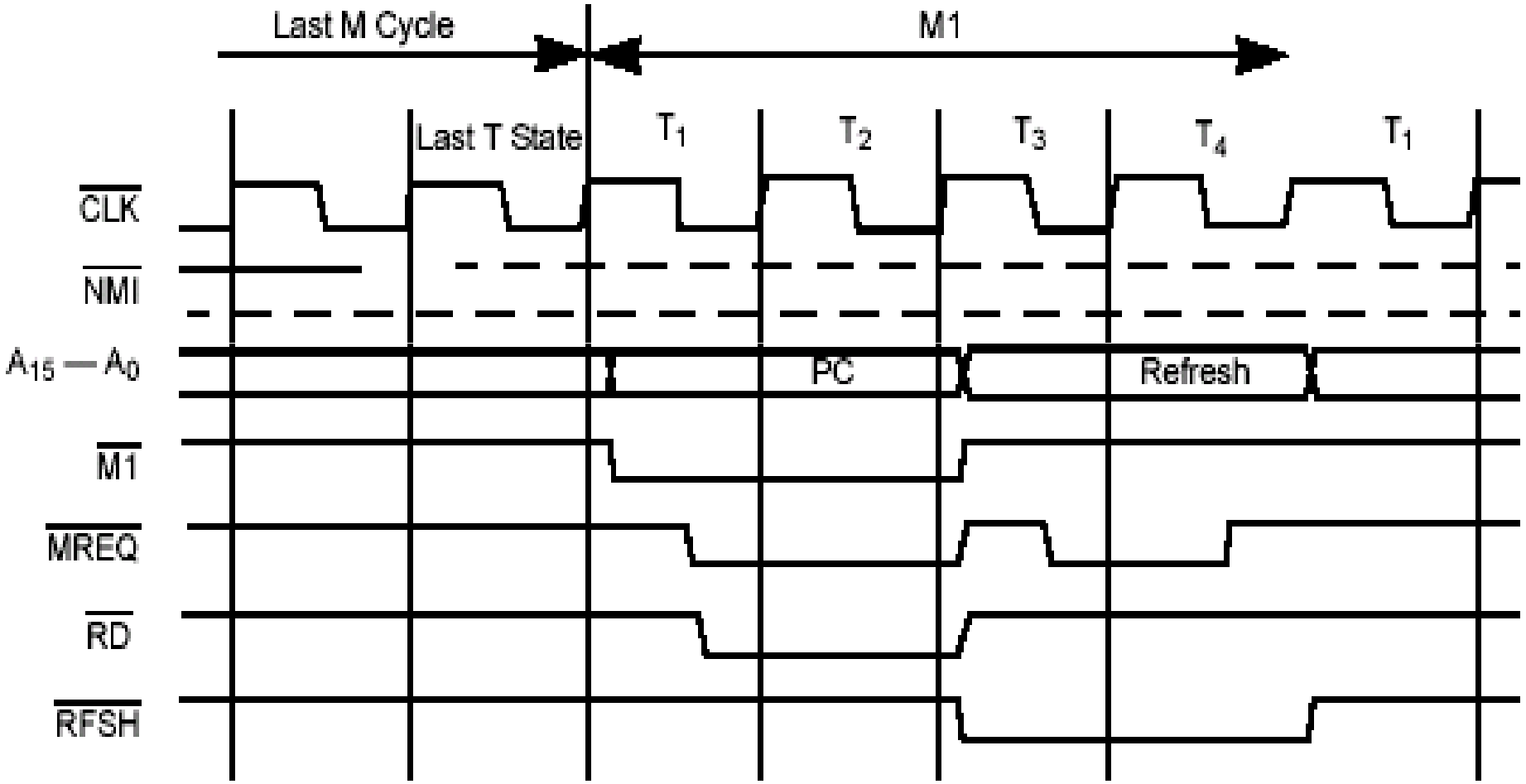


Chu kỳ yêu cầu/ghi nhận ngắt



Two wait states are automatically added to this cycle

Chu kỳ yêu cầu/ghi nhận bus với NMI



Chu kỳ làm tươi M1

- Takes 4T to 6Ts
- Z80 includes built in circuitry for refreshing DRAM
- This simplifies the external interfacing hardware
- DRAM consists of MOS transistors, which store Information as capacitive charges; each cell needs to be periodically refreshed
- During T3 and T4 (when Z80 is performing internal ops), the low order address is used to supply a 7-bit address for refresh

Tín hiệu Wait

- the Z80 samples the wait signal during T2 if low then Z80 adds wait
- states to extend the machine cycle
- used to interface memories with slow response time
- Slow memory is low cost

Ngắt (Interrupt)

There are two types of interrupts:

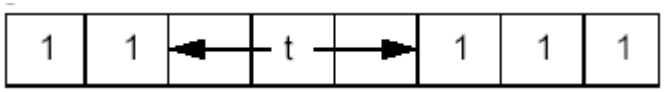
- non mask-able (NMI)
  - Could not be masked
  - Jump to 0066H of memory
- mask-able(INT)
  - Has 3 mode
  - Can be set with the IM x Instruction
  - IM 0 sets Interrupt mode 0
  - IM 1 sets Interrupt mode 1
  - IM 2 sets Interrupt mode 2

Các chế độ ngắt

- Mode 0:
  - An 8 bit opcode is Fetched from Data BUS and executed
  - The source interrupt device must put 8 bit opcode at data bus
  - 8 bit opcode usually is RST p instructions
- Mode 1:
  - A jump is made to address 0038h
  - No value is required at data bus
- Mode 2:
  - A jump is made to address (register I × 256 + value from interrupting device that puts at bus)
  - I is high 8 bit of interrupt vector
  - Value is low 8 bit of interrupt vector

RST p

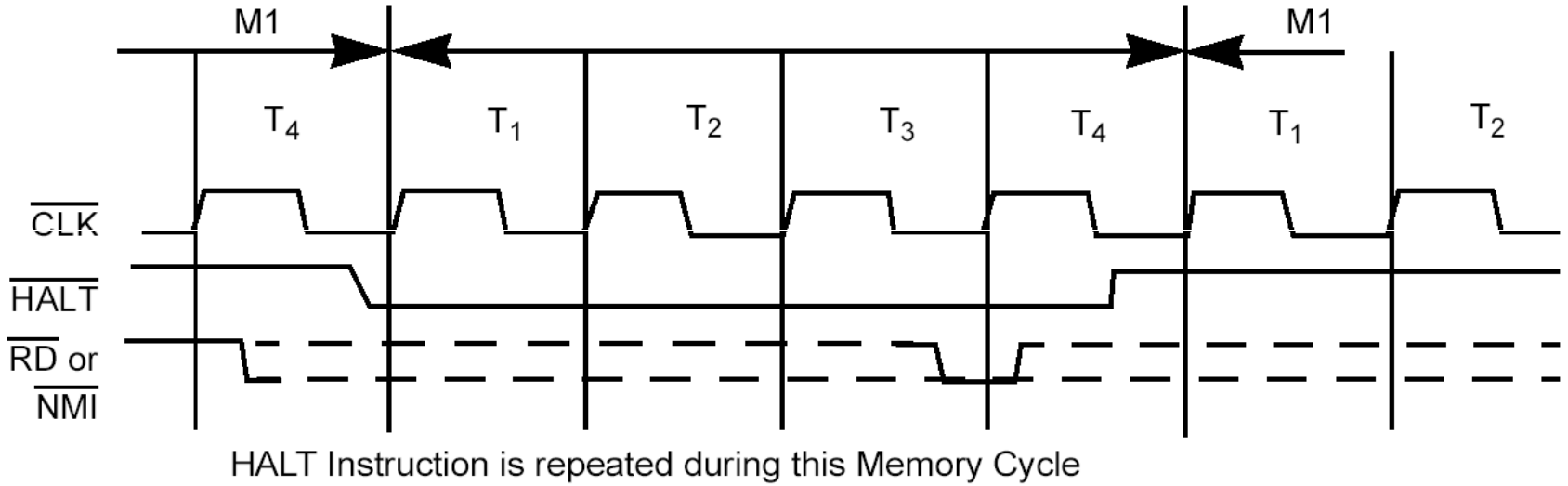
Operation: (SP-1) ← PCH, (SP-2) ← PCL, PCH ← 0, PCL ← P



p	t
00H	000
08H	001
10H	010
18H	011
20H	100
28H	101
30H	110
38H	111

		Op Code	
CALL Address	0000H	C7	RST 0
	0008H	CF	RST 8
	0010H	D7	RST 16
	0018H	DF	RST 24
	0020H	E7	RST 32
	0028H	EF	RST 40
	0030H	F7	RST 48
	0038H	FF	RST 56

Chu kỳ thoát khỏi HALT



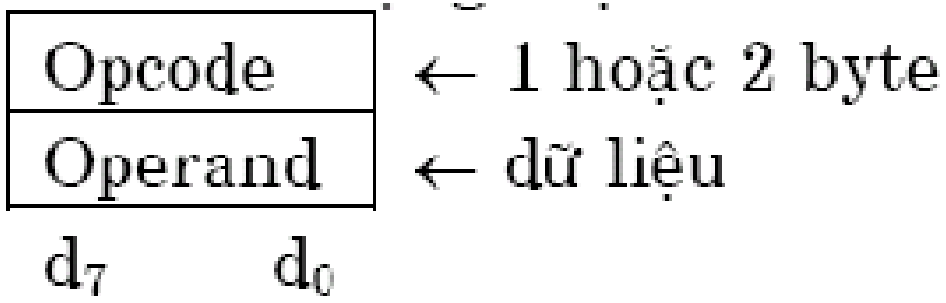
2.6 CÁC PHƯƠNG PHÁP ĐỊNH ĐỊA CHỈ

Định địa chỉ trong Z80

- Phần lớn các lệnh Z80 làm việc với dữ liệu được lưu trữ trong các thanh ghi CPU, bộ nhớ ngoài, hoặc trong các cổng I/O. Z80 có cách định địa chỉ sau:
  - ***Định địa chỉ tức thời***
  - ***Định địa chỉ tức thời mở rộng***
  - ***Định địa chỉ trực tiếp***
  - ***Định địa chỉ trang 0 (có sửa đổi)***
  - ***Định địa chỉ tương đối***
  - ***Định địa chỉ mở rộng***
  - ***Định địa chỉ theo chỉ số***
  - ***Định địa chỉ thanh ghi***
  - ***Định địa chỉ hiểu ngầm***
  - ***Định địa chỉ gián tiếp qua thanh ghi***
  - ***Định địa chỉ cho bit***

Định địa chỉ tức thời

- Byte theo sau mã lệnh là toán hạng thật.

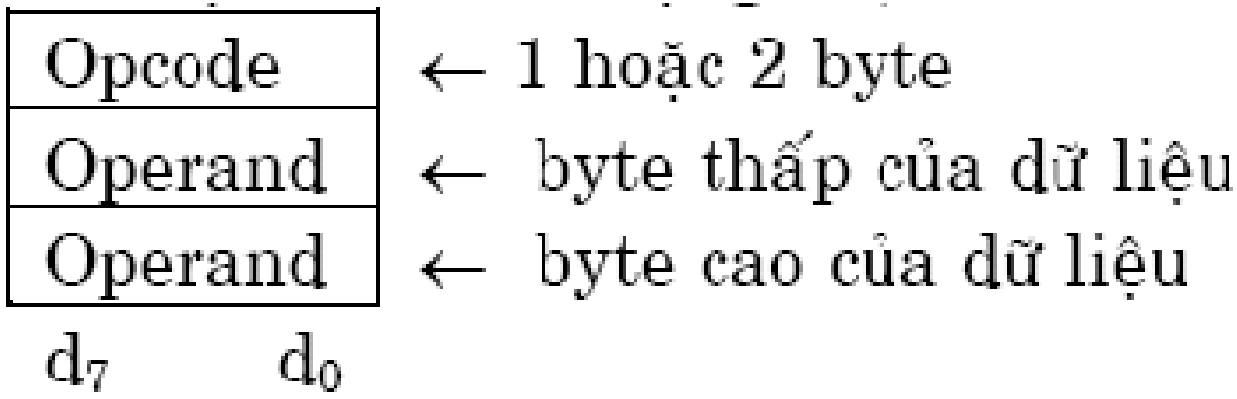


- Thí dụ của loại lệnh này là nạp hằng số vào thanh ghi tích lũy.

Thí dụ: LD A, 10H

Định địa chỉ tức thời mở rộng

- Hai byte theo sau mã lệnh là toán hạng thật.



- Thí dụ của loại lệnh này là nạp dữ liệu 16 bit vào cặp thanh ghi (thí dụ HL).

Thí dụ: LD HL, 8010H

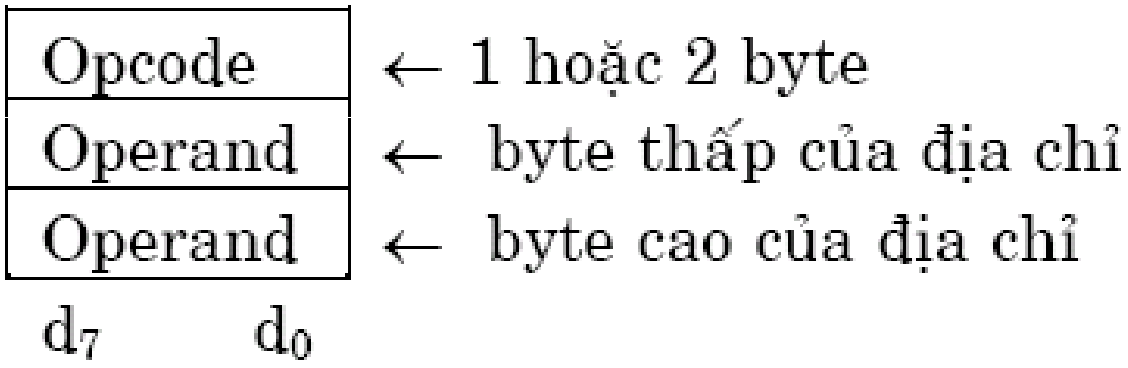
Định địa chỉ trực tiếp

- Cung cấp địa chỉ 8 bit của toán hạng ngay sau mã lệnh

- TD: IN A, (20H)

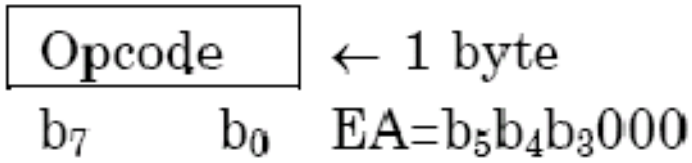
Định địa chỉ trực tiếp mở rộng

- Cung cấp địa chỉ 16 bit của toán hạng ngay sau mã lệnh



Định địa chỉ trang 0 (có sửa đổi)

- Z80 có lệnh CALL đặc biệt 1 byte để nhảy đến 8 vị trí (chỉ bởi nhóm bit $b_5b_4b_3$) của trang 0 của bộ nhớ. Lệnh này được thực thi như khởi động lại, nó đặt PC có giá trị địa chỉ thật ở trang 0. Giá trị của lệnh này là cho phép dùng 1 byte để chỉ địa chỉ 16 bit.



- Thí dụ: RST p

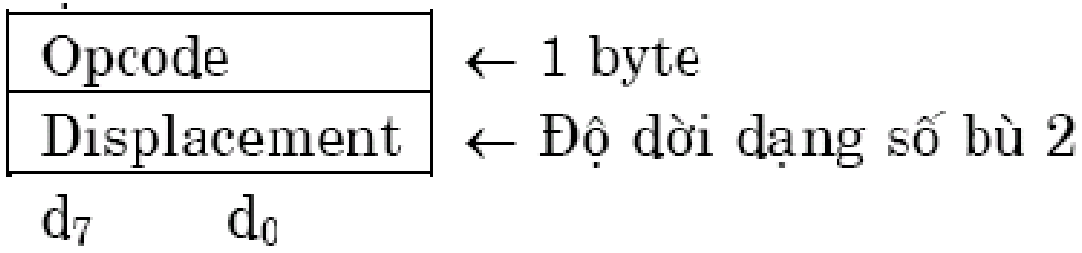
  với giá trị p có thể là 00H, 08H, 10H, 18H, 20H, 28H, 30H, hoặc 38H.

Định địa chỉ tương đối

- Sử dụng 1 byte dữ liệu theo sau mã lệnh để chỉ độ dời so với địa chỉ lệnh kế và lệnh định nhảy đến. Độ dời D này số có dấu biểu diễn theo số bù 2 và địa chỉ thật được tính như sau (theo độ dời D và địa chỉ lệnh hiện tại A):

$$EA = D + A + 2$$

Độ dời D có thể có giá trị từ -128 đến +127.



- Thí dụ: JR 10H

Định địa chỉ theo chỉ số

- Byte dữ liệu theo mã lệnh là độ dời D (số có dấu bù 2) được cộng vào với thanh ghi chỉ số (IX hoặc IY) để chỉ đến ô nhớ dữ liệu, nghĩa là EA = IX (hoặc IY) + D. Loại lệnh này có mã lệnh dài 2 byte và trong mã lệnh có các bit chỉ ra thanh ghi chỉ số nào được chọn IX hoặc IY.
- Thí dụ: LD A, (IX + 10H)

  LD B, (IY + 20H)

Định địa chỉ thanh ghi

- Nhiều mã lệnh Z80 chứa các bit thông tin cho biết thanh ghi CPU nào được sử dụng trong lệnh.
- Thí dụ: LD A, B

Định địa chỉ hiểu ngầm

- Mã lệnh cho biết 1 hay nhiều thanh ghi CPU chứa toán hạng. Thí dụ có một số lệnh hiểu toán hạng để ở thanh ghi tích lũy.

Định địa chỉ gián tiếp qua thanh ghi

- Loại định địa chỉ này cho biết cặp thanh ghi 16 bit nào (như HL) được dùng làm con trỏ chỉ tới vị trí ô nhớ.
- Thí dụ:

```
LD   A, (HL)
LD   (HL), 10H
```

Định địa chỉ cho bit

- Z80 có nhiều lệnh đặt bit, xóa bit và kiểm tra bit. Các lệnh này cho phép bất cứ vị trí bộ nhớ nào hoặc thanh ghi CPU sẽ được sử dụng cho các phép toán bit qua một trong 3 cách định địa chỉ trước (thanh ghi, gián tiếp qua thanh ghi và theo chỉ số) và 3 bit trong mã lệnh sẽ cho biết bit nào trong 8 bit được xử lý.

  Thí dụ:

```
BIT   3, A
SET   0, (HL)
RES   7, (IX + 10H)
```

  Chú ý là có những lệnh kết hợp một số cách địa chỉ chung trong một lệnh.

2.7 TẬP LỆNH

- Tập lệnh Z80 gồm có 158 lệnh, trong đó đã bao gồm 78 lệnh của 8080. Các lệnh được chia làm các nhóm chính sau:
  - Nạp 8 bit và nạp 16 bit
  - Hoán đổi, chuyển khối và tìm kiếm
  - Số học và logic 8 bit
  - Số học 16 bit
  - Số học đa dụng và điều khiển CPU
  - Xoay và dịch bit
  - Xử lý bit (Các phép toán trên bit)
  - Nhập và xuất
  - Nhảy, rẽ nhánh chương trình
  - Gọi chương trình và quay về chương trình gọi.

- Khi viết chương trình để thuận tiện thường ta dùng hợp ngữ mà lệnh có dạng như trong thí dụ sau:

Label (Nhãn chương trình)	Opcode (mã lệnh với từ gợi nhớ [mnemonic])	Operand/Address (Toán hạng hoặc địa chỉ)	Comment (Chú thích)
LP:	ADD	A, A	; 2 x A
	DEC	B	
	HALT		

Pseudo Codes

- Để có thể viết chương trình dễ dàng hơn các trình hợp ngữ thường cho thêm các tác vụ giả (pseudo operation) hay còn gọi là các lệnh giả như: EQU, DEFB, DEFW, DEFS, ORG, END, MACRO.

1. **EQU** (nghĩa là equate=bằng; dùng để gán trị)
   - Thí dụ: Nội dung ô nhớ SUM = A + 10H

```
SUM:      EQU    1900H        ; SUM = 1900H
COUNT:    EQU    10H
          ADD A, COUNT        ; ADD A, 10H
          LD (SUM), A
```

2. **DEFB** (Define Byte = Định nghĩa Byte) và DEFW (Define Word = Địnhnghĩa Word)

Thí dụ:

```
CONST:    EQU     52H
ADRS:     EQU     1900H
TABLE:    DEFW    ADRS
          DEFB    CONST
```

Khi đó các vùng nhớ (giả sử địa chỉ ở TABLE là 1951H) như sau

Địa chỉ	Dữ liệu
1951H	00H
1952H	19H
1953H	52H

3. **DEFS** (Define Storage = Định nghĩa vùng lưu trữ)

Thí dụ: (Giả sử lệnh DEFS 10H ở địa chỉ 1800H)

```
1800H          DEFS  10H ; dành 16 byte lưu trữ
1811H          LD    HL, 10H
1812H          LD    DE, 20H
```

4. ORG (Origin = bắt đầu; định nghĩa địa chỉ bắt đầu) và **END** (kết thúc chương trình)

Thí dụ:

```
          ORG     100H ; Chương trình bắt đầu từ 100H
START:    LD      A, 1FH ; bắt đầu chương trình
          INC     A
          OUT     (11H), A
          END             ; kết thúc chương trình
```

5. MACRO (Định nghĩa một đoạn chương trình mà trình hợp ngữ sẽ tự động chèn vào toàn bộ đoạn chương trình này khi có tham chiếu tên macro đó và bắt đầu đoạn macro bằng MACRO và kết thúc bằng **ENDM**)

Thí dụ:

```
REF:    MACRO
        LD      H, (IX+1)
        LD      L, (IY+1)
        INC     IX
        INC     IY
        ENDM
```

- ***Thí dụ:*** Điền các số 0 vào các ô nhớ có địa chỉ từ 8100H đến 817FH.

```
        ORG 8000H
        LD   HL, 8100H
        LD   A, 00H
        LD   C, 80H
LOOP:   LD   (HL), A
        INC  HL
        DEC  C
        JP   NZ, LOOP
        HALT
        END
```

- ***Thí dụ:*** Nhân 10 lần giá trị được cất trong cặp thanh ghi HL.

```
MULT10:   ADD  HL, HL     ; 2X HL
          LD   D, H       ; (HL)=>(DE)
          LD   E, L
          ADD  HL, HL     ; 4 x HL
          ADD  HL, HL     ; 8 x HL
          ADD  HL, DE     ; (8+2) x HL
          RET
```

- ***Thí dụ:*** Chuyển chuỗi dữ liệu 737 byte ở vị trí bộ nhớ DATA đến vị trí bộ nhớ BUFFER, tác vụ này được lập trình như sau

```
LD HL, DATA       ; Lấy địa chỉ bắt đầu của DATA
LD DE, BUFFER     ; Lấy địa chỉ bắt đầu của BUFFER
LD BC, 737        ; Chiều dài của chuỗi ký tự
LDIR              ; Chuyển chuỗi–Chuyển bộ nhớ chỉ
                  ; bởi HL vào ô nhớ được chỉ bởi DE
                  ; tăng  HL và DE lên 1
                  ;Giảm bớt BC đi 1, cho đến khi BC= 0.
```

Transfer & Exchange

- LD *dst*, *src*
- LD *reg*, (*addr*)
- LD *reg*, *data*
- LD (*addr*), *reg*
- EX DE, HL
- EX AF, AF'
- EXX

Arithmetic

- ADD A, *data*
- ADD A, *reg*
- ADD A, (*addr*)
- ADD *rp*, *rp*
- ADC A, *data*
- ADC A, (*addr*)
- ADC A, *reg*
- ADC *rp*, *rp*
- SUB *data*
- SUB *reg*
- SUB (*addr*)
- SBC *data*
- SBC A, *reg*
- SBC A, (*addr*)
- SBC *rp*, *rp*
- INC *reg*
- INC (*addr*)
- DEC *reg*
- DEC (*addr*)

Logic

- AND *reg*
- AND *data*
- AND (*addr*)
- OR *reg*
- OR *data*
- OR (*addr*)
- XOR *reg*
- XOR *data*
- XOR (*addr*)
- CP *reg*
- CP *data*
- CPI
- CPIR
- CPD
- CPDR

Control

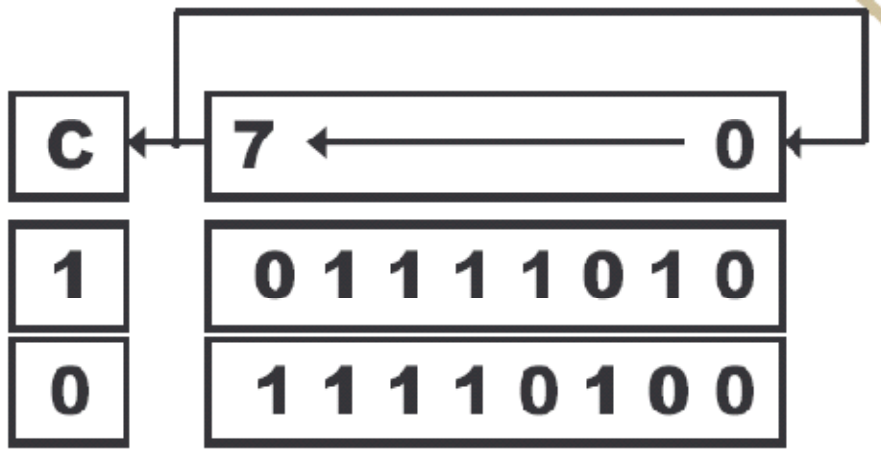
- DI
- EI
- IM 0,1 or 2
- NOP
- HLT

Accumulator & Flags

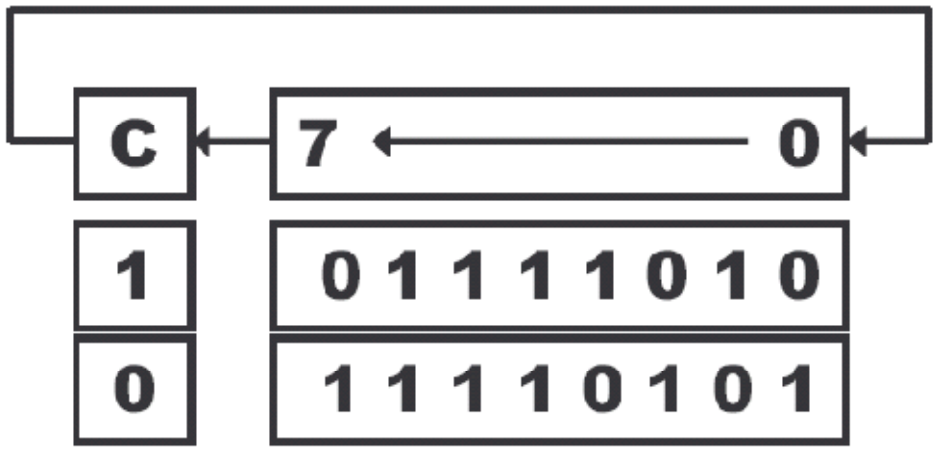
- DAA
- CPL
- NEG
- SCF
- CCF

Rotate & Shift

- RLCA
- RLC *reg* or (*addr*)
  - before:
  - after:

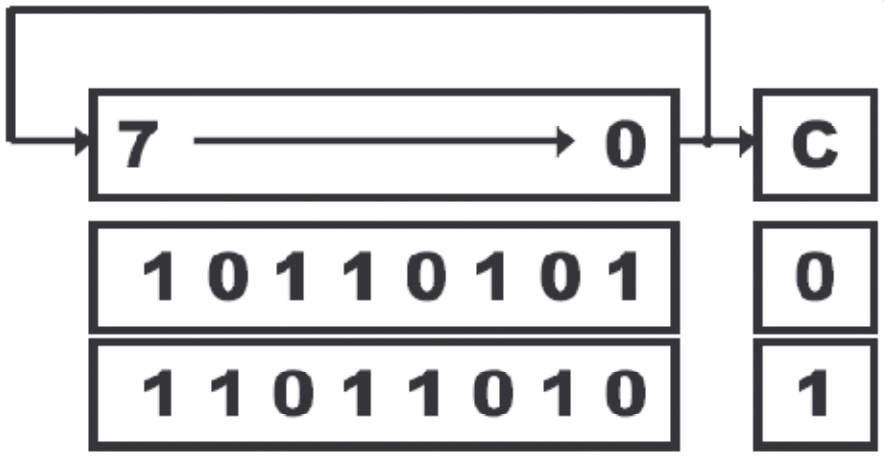


- RLA
- RL *reg* or (*addr*)
  - before:
  - after:

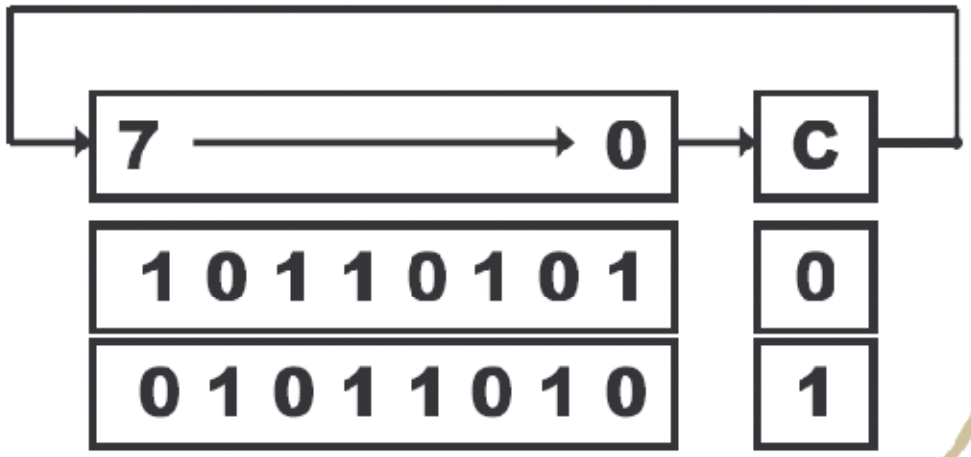


Rotate & Shift (cont.)

- RRCA
- RRC *reg* or (*addr*)
  - before
  - after



- RRA
- RR *reg* or (*addr*)
  - before
  - after



Rotate & Shift (cont.)

- RLD
- RRD
- SLA *reg*
- SLA (*addr*)
- SRA *reg*
- SRA (*addr*)
- SRL *reg*
- SRL (*addr*)

Branch Control

- JP *addr*
- JP *cond*, *addr*
- JR *displacement*
- JR cond, *addr*
- CALL *addr*
- CALL *cond*, *addr*
- RET
- RET *cond*
- RETI
- RETN
- RST

Stack

- PUSH rp
- PUSH 2Br
- POP rp
- POP 2Br
- EX (SP), *rp*

Input & Output

- IN *reg*, (*port*)
- INI
- INIR
- IND
- INDR
- OUT (*port*), *reg*
- OUTI
- OTIR
- OUTD
- OTDR

Bit Manipulation

- BIT (0-7), *reg*
- BIT (0-7), (*addr*)
- RES (0-7), *reg*
- RES (0-7), (*addr*)
- SET (0-7), *reg*
- SET (0-7), (*addr*)