THỰC HÀNH LẬP TRÌNH HỢP NGỮ TRÊN 8086

Trước khi vào thực hành:

- Nắm được mục tiêu của bài thực hành.
- Xem lại các kiến thức cần chuẩn bị được nêu ra cho mỗi bài thực hành.
- Nắm được các nội dung cần phải làm trong buổi thực hành.

Trong khi thực hành:

- Tuyệt đối tuân thủ thực hành theo thứ tự của nội dung thực hành. Hoàn thành các vấn đề và trả lời được các câu hỏi đặt ra trong phần trước mới chuyển sang thực hành phần sau.
- Quan sát hiện tượng, những thay đổi, xem xét đánh giá kết quả sau mỗi thao tác thực hành.
- Lập lại các thao tác thực hành nhiều lần, tìm cách giải quyết khác sau khi đã thực hành theo yêu cầu cho mỗi vấn đề. So sánh, nhận xét các cách giải quyết khác nhau.

Sau khi thực hành:

- Đối chiếu từng mục tiêu của bài thực hành với những gì đã thực hành được. Nếu mục tiêu nào chưa thành thạo thì phải tìm cách lập lại thực hành đó để nắm được mục tiêu vững chắc hơn.

NỘI DUNG

1. Giới thiệu về 8086 và lập trình hợp ngữ
2. Nhập xuất ký tự
3. Nhập xuất số BIN, HEX, DEC
4. Cấu trúc rẽ nhánh- vòng lặp
5. Xử lý tập tin
6. Xử lý chuỗi ký tự
7. Các bài nâng cao

NỘI DUNG 1

GIỚI THIỆU 8086 VÀ LẬP TRÌNH HỢP NGỮ

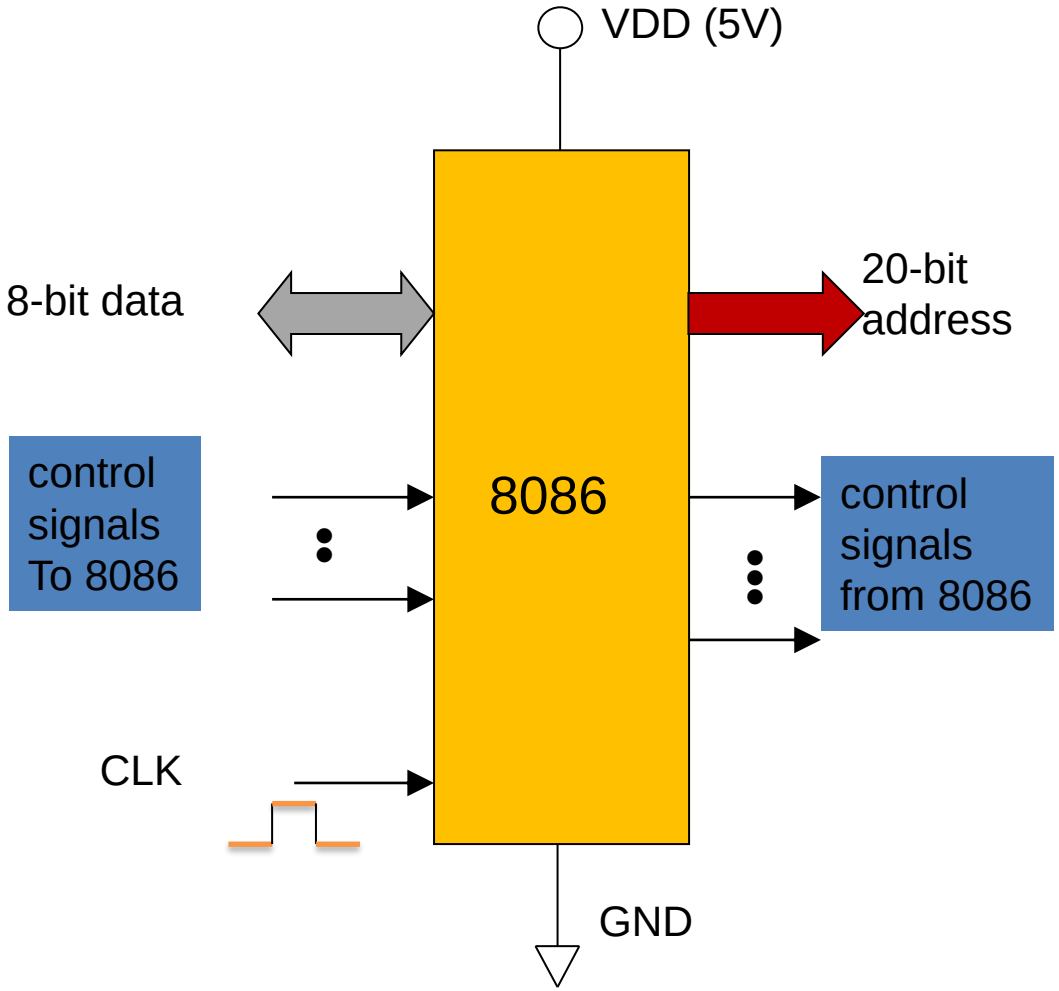
Mục tiêu: Hiểu được kiến trúc của 8086 và cấu trúc của chương trình hợp ngữ, công cụ Emu8086 để lập trình trên máy tính.

1.1. Tổng quan 8086

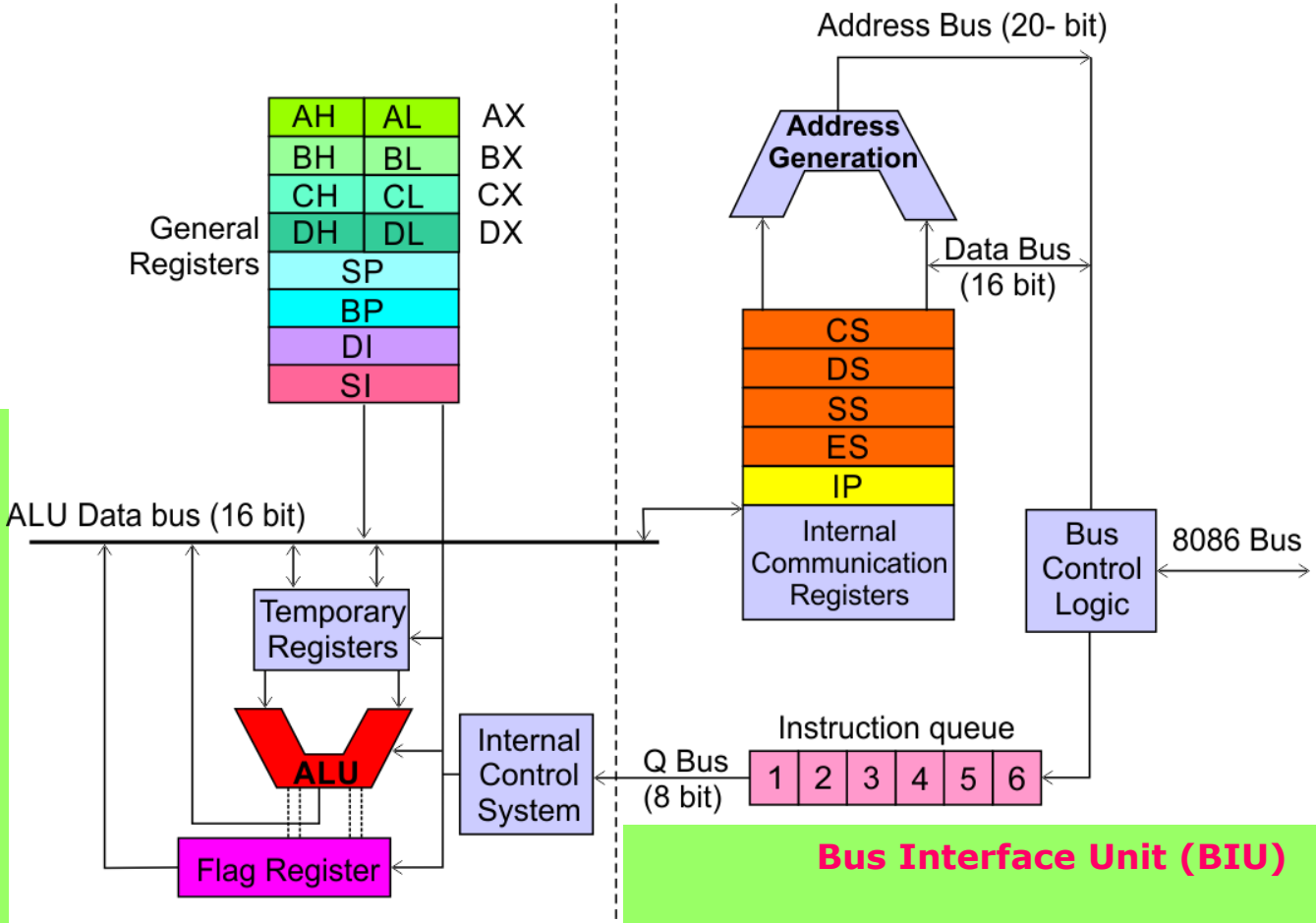
Là bộ vi xử lý thuộc thế hệ thứ 3. Bộ vi xử lý 8086 xử lý dữ liệu 16 bit của Intel, các thông số cơ bản của 8086 như sau:

- Được đóng trong vỏ 40 chân
- Nguồn nuôi: chỉ một loại nguồn +5V
- Có khả năng xử lý dữ liệu **8 bit hay 16 bit**.
- Cả bus dữ liệu bên trong và ngoài đều là 16 bit
- Bus địa chỉ 20 bit, bộ vi xử lý sử dụng cả **20 bit để địa chỉ hoá bộ nhớ**, do vậy khả năng địa chỉ hoá tối đa là **1MB bộ nhớ**.
- Với cổng vào/ra, bộ vi xử lý sử dụng 8 bit địa chỉ trong chế độ địa chỉ trực tiếp, và 16 bit địa chỉ trong chế độ địa chỉ gián tiếp.
- Có khả năng làm việc song song giữ **đơn vị điều khiển bus** (BIU) và **đơn vị xử lý** (EU).
- Có thể làm việc ở 2 chế độ: chế độ tối thiểu (**MIN**), chế độ mở rộng (**MAX**).

1.1. Tổng quan 8086



1.2. KIẾN TRÚC BÊN TRONG 8086



Execution Unit (EU)

EU executes instructions that have already been fetched by the BIU.

BIU and EU functions separately.

Bus Interface Unit (BIU)

BIU fetches instructions, reads data from memory and I/O ports, writes data to memory and I/ O ports.

1.2. KIẾN TRÚC BÊN TRONG 8086

Đơn vị giao tiếp bus (BIU)

❖ ***Chức năng:*** giao tiếp với các thành phần bên ngoài (bộ nhớ, cổng vào/ra) thông qua bus hệ thống. BIU thực hiện các chức năng:

- Tạo địa chỉ vật lý từ địa chỉ logic của ô nhớ cần truy xuất;
- Đọc mã lệnh từ bộ nhớ đưa tới hàng đợi lệnh;
- Truy xuất bộ nhớ hay cổng vào/ra để đọc hoặc gửi dữ liệu.

1.2. KIẾN TRÚC BÊN TRONG 8086

2.3.2. Đơn vị giao tiếp bus (BIU)

❖ ***Các khối:***

- Bộ cộng: tạo địa chỉ vật lý
- 4 thanh ghi đoạn **16 bit CS, DS, SS, ES**; một thanh ghi con trỏ lệnh **16 bit IP**
- Hàng đợi lệnh: dùng để lưu trữ tạm thời các byte lệnh sẽ được đưa vào thực hiện trong EU. Hàng đợi lệnh được cấu tạo theo kiểu FIFO. Hàng đợi lệnh của 8086 là 6 byte.
- Khối điều khiển Bus: Khối điều khiển bus (bus control unit) thực hiện việc ghép nối giữa CPU với Bus hệ thống.

1.2. KIẾN TRÚC BÊN TRONG 8086

2.3.3. Đơn vị thực hiện lệnh EU (Execution Unit)

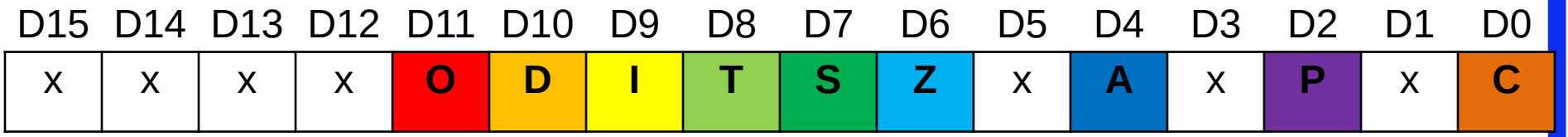
❖ ***Chức năng:*** EU nhận mã lệnh và dữ liệu từ BIU, thực thi lệnh. Kết quả thực thi lệnh được chuyển ra bộ nhớ hoặc cổng vào/ra thông qua BIU.

❖ ***Các khối:***

- Các thanh ghi tạm thời: lưu mã lệnh, dữ liệu tạm thời tạm thời. Mã lệnh sau đó được đưa tới bộ giả mã để xác định lệnh.
- ALU: thực hiện các phép toán số học và logic
- Điều khiển EU: điều khiển quá trình thực hiện lệnh.
- 8 thanh ghi 16 bit: **AX, BX, CX, DX, SP, BP, SI, DI** và một thanh ghi cờ (**FR**) (chức năng của các thanh ghi này sẽ đề cập sau)

1.3. CÁC THANH GHI

- Các thanh ghi đoạn: DS, CS, ES, SS (16 bit)
- Các thanh ghi đa năng: AX, BX, CX, DX (16 bit)
- Các thanh ghi con trỏ và chỉ số: IP, BP, SI, DI (16 bit)
- Thanh ghi cờ: 16 bit, nhưng có 9 bit được sử dụng



1.3. CÁC THANH GHI

Thanh ghi cờ

Cờ trạng thái: có 6 cờ trạng thái là **AF, CF, SF, PF, ZF, OF**: phản ánh các trạng thái khác nhau của kết quả sau một thao tác nào đó.

CF (Carry flag): cờ nhớ chính, CF=1 khi có nhớ hoặc mượn từ bit MSB.

AF (Axuxiliary flag): cờ nhớ phụ, rất có ý nghĩa khi làm việc với các số BCD. AF=1 khi có nhớ hoặc mượn từ một số BCD thấp (4 bit thấp) sang một số BCD cao (4 bit cao).

SF (Sign flag): cờ dấu, SF=1 khi kết quả âm.

PF (Parity flag): cờ chẵn lẻ, PF =1 khi số các số 1 ở kết quả phép toán là chẵn

ZF (Zero flag): cờ Zero, ZF=1 khi kết quả phép toán bằng 0

OF (Over flag): cờ tràn, OF=1 khi kết quả là một số bù 2 vượt ra ngoài giới hạn biểu diễn dành cho nó.

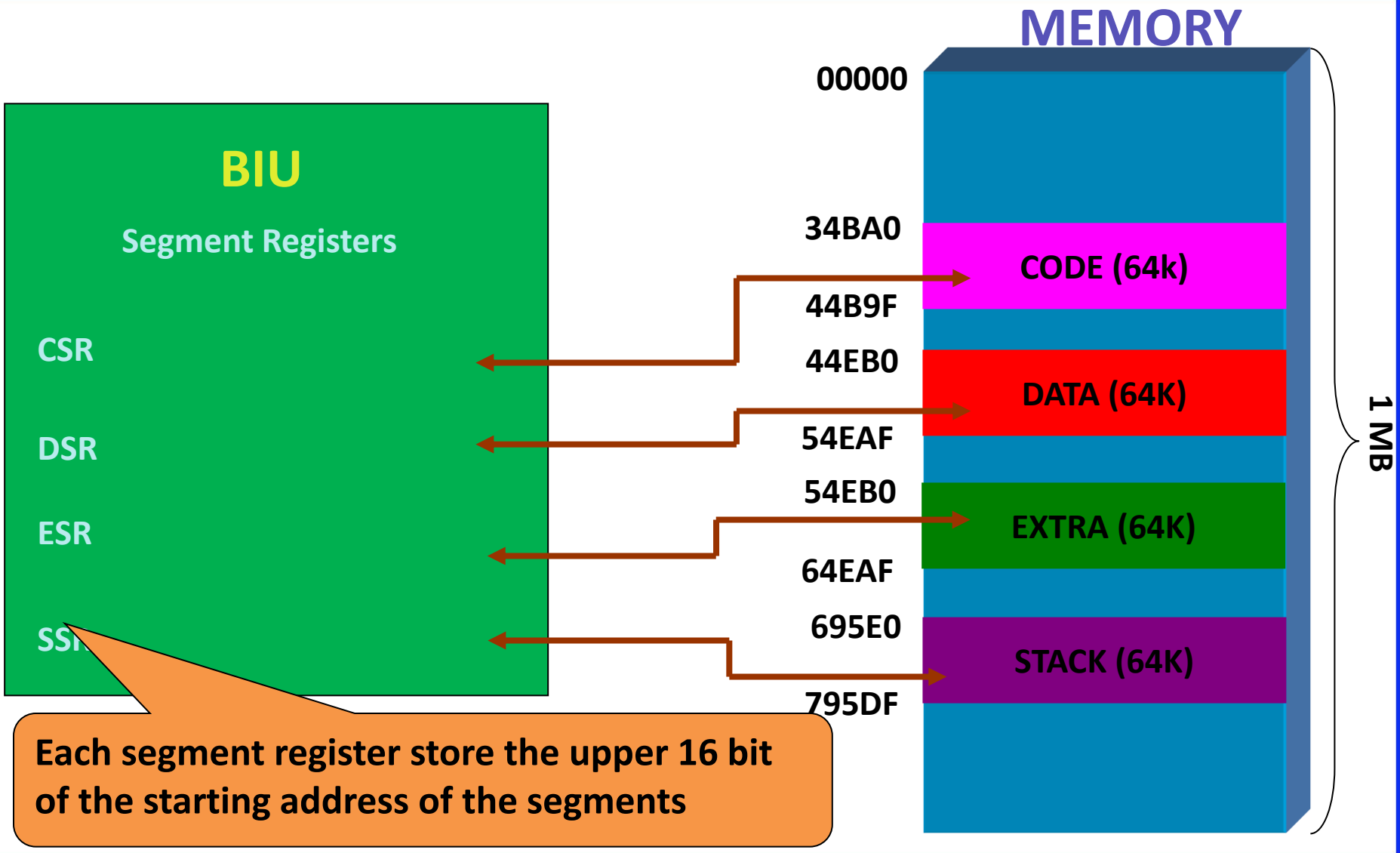
Cờ điều khiển: có 3 cờ điều khiển là **IF, TF, DF**. Các cờ này được lập xóa bằng các lệnh riêng

IF (Interrupt flag): cờ ngắt, khi IF=1 CPU cho phép các yêu cầu ngắt (che được) được tác động.

TF (Trap flag): cờ bẫy, TF=1 thì CPU sẽ ở chế độ chạy từng lệnh, giúp cho việc hiệu chỉnh và gỡ rối chương trình.

DF (Direction flag): cờ hướng, DF=1 khi CPU làm việc với chuỗi ký tự theo thứ tự từ phải qua trái (vì vậy DF chính là cờ lùi).

1.3. CÁC THANH GHI



Each segment register store the upper 16 bit of the starting address of the segments

1.4. CHẾ ĐỘ ĐỊA CHỈ CỦA 8086

Các chế độ địa chỉ giúp chúng ta hiểu các loại toán hạng và cách chúng được truy cập trong khi thực hiện một lệnh.

1.4. CHẾ ĐỘ ĐỊA CHỈ CỦA 8086

1. Immediate addressing mode (Chế độ địa chỉ tức thì)
2. Direct addressing mode (Chế độ địa chỉ trực tiếp)
3. Register addressing mode (Chế độ địa chỉ thanh ghi)
4. Register Indirect addressing mode (Chế độ địa chỉ gián tiếp qua thanh ghi)
5. Indexed addressing mode (Chế độ địa chỉ chỉ số)
6. Register relative addressing mode (Chế độ địa chỉ tương đối thanh ghi)
7. Base plus index addressing mode (Chế độ địa chỉ chỉ số, cơ sở)
8. Base relative plus index addressing mode (Chế độ địa chỉ tương đối chỉ số, cơ sở)

1.4. CHẾ ĐỘ ĐỊA CHỈ CỦA 8086

❖ **Bỏ ngầm định thanh ghi đoạn:**

Đó là những thanh ghi đoạn đã được ngầm định đối với những offset được cho trong lệnh khi xác định địa chỉ ô nhớ, được tổng kết trong bảng sau:

1.4. CHẾ ĐỘ ĐỊA CHỈ CỦA 8086

2.5.6. Các chế độ địa chỉ của 8086

Chế độ địa chỉ	*Toán hạng*	*Thanh ghi ngầm định*
Trực tiếp	[offset]	DS
Gián tiếp qua thanh ghi	[BX]	DS
	[SI]	DS
	[DI]	DS
Tương đối cơ sở	[BX]+Disp	DS
	[BP]+Disp	SS
Tương đối chỉ số	[SI]+Disp	DS
	[DI]+Disp	DS
Tương đối chỉ số cơ sở	[BX]+[DI]+Disp	DS
	[BX]+[SI]+Disp	DS
	[BP]+[DI]+Disp	SS
	[BP]+[SI]+Disp	SS

1.4. CHẾ ĐỘ ĐỊA CHỈ CỦA 8086

1. Immediate addressing mode

- Trong loại chế độ này, dữ liệu tức thời là một phần của lệnh và xuất hiện dưới dạng 1 byte hoặc 2 byte liên tiếp

10 AB_H

MOV AX,$10AB_H$

AX

1.4. CHẾ ĐỘ ĐỊA CHỈ CỦA 8086

1. Immediate addressing mode

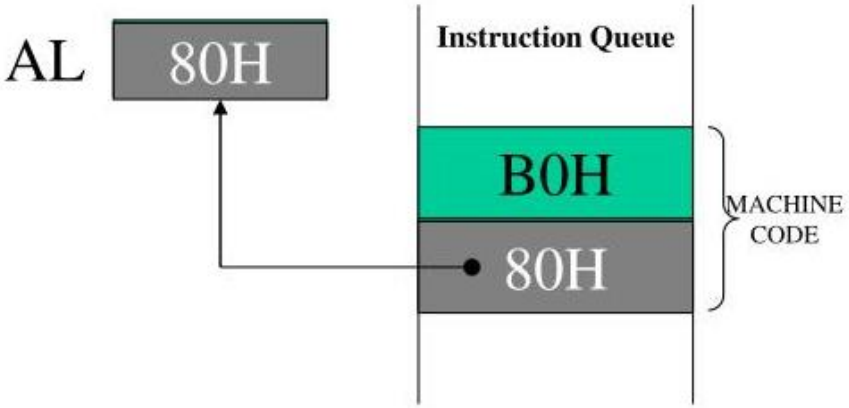
MOV AL, 80H

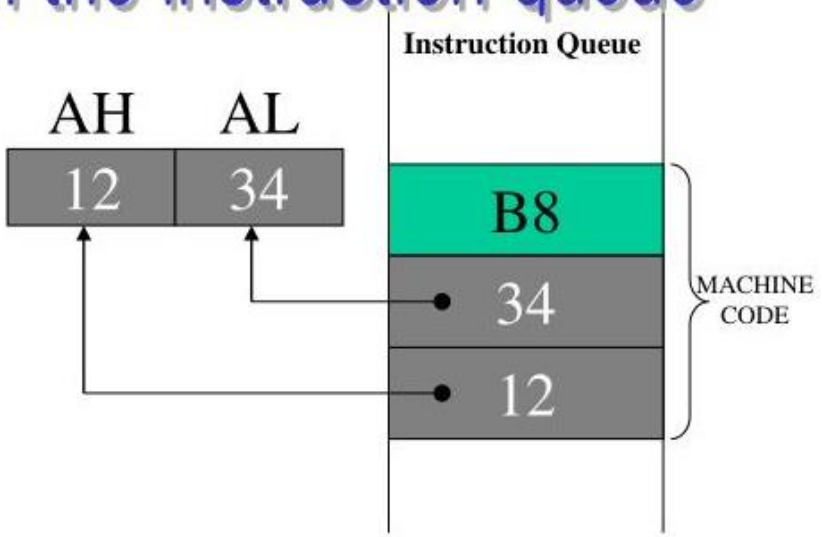
Machine code:B080H

MOV AX, 1234H

Machine code:B83412H

CPU gets the operand from the instruction queue





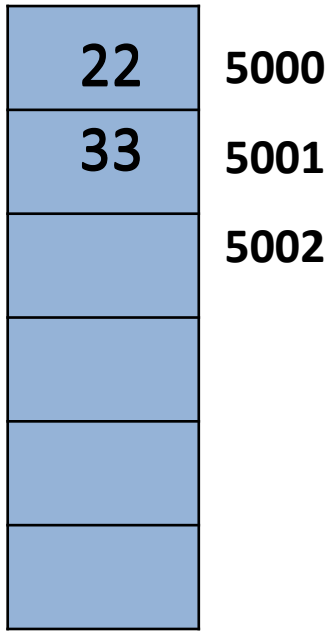
1.4. CHẾ ĐỘ ĐỊA CHỈ CỦA 8086

2. Direct addressing mode

- Trong loại chế độ địa chỉ này, truy cập trực tiếp địa chỉ ô nhớ 16 bit.

MOV AX,[5000H]

AX

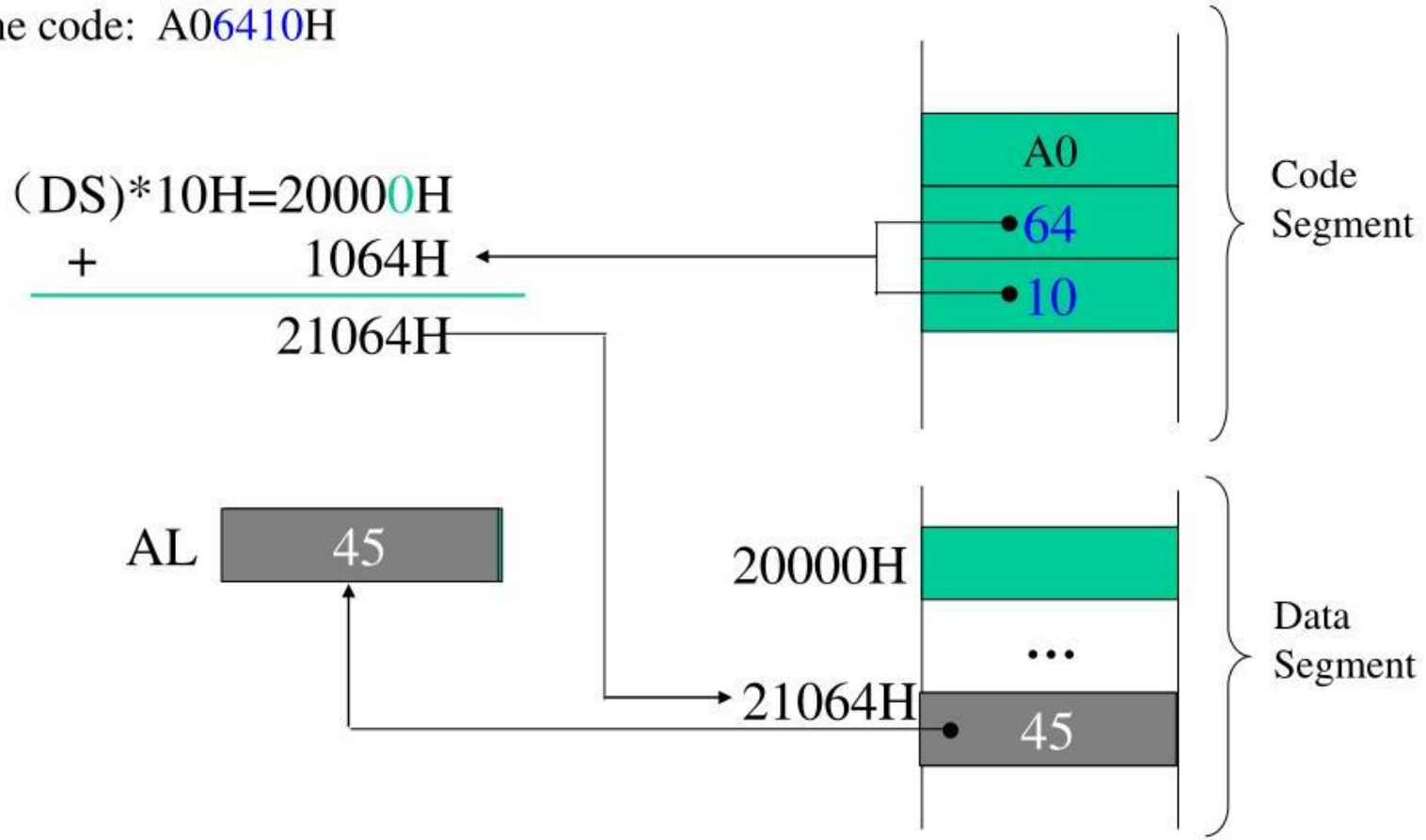


1.4. CHẾ ĐỘ ĐỊA CHỈ CỦA 8086

2. Direct addressing mode

Example: MOV AL, [1064h] ;Assume (DS)=2000H

Machine code: A06410H



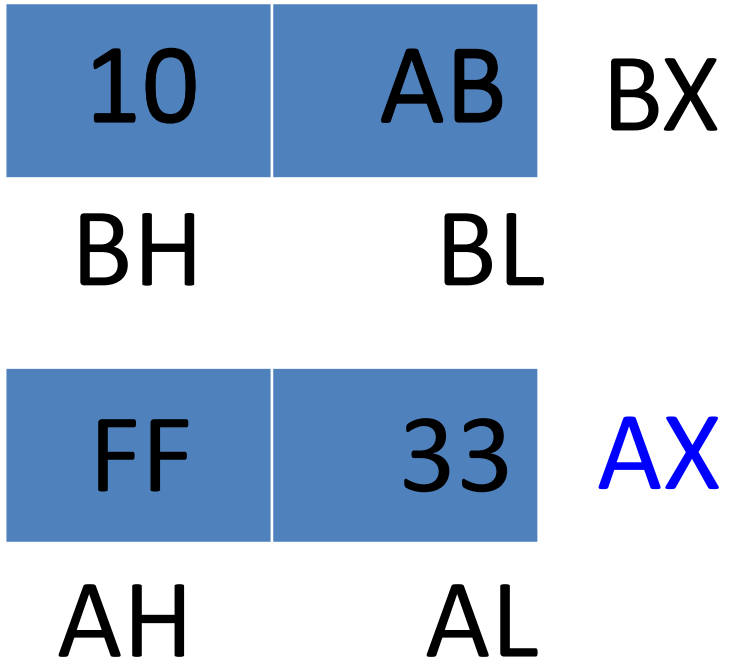
1.4. CHẾ ĐỘ ĐỊA CHỈ CỦA 8086

3. Register addressing mode

- Trong loại chế độ địa chỉ này, dữ liệu được lưu trữ trong thanh ghi và nó có thể là thanh ghi 8 bit hoặc 16 bit. Tất cả các thanh ghi, ngoại trừ IP, có thể được sử dụng trong chế độ này.

MOV AL,BL

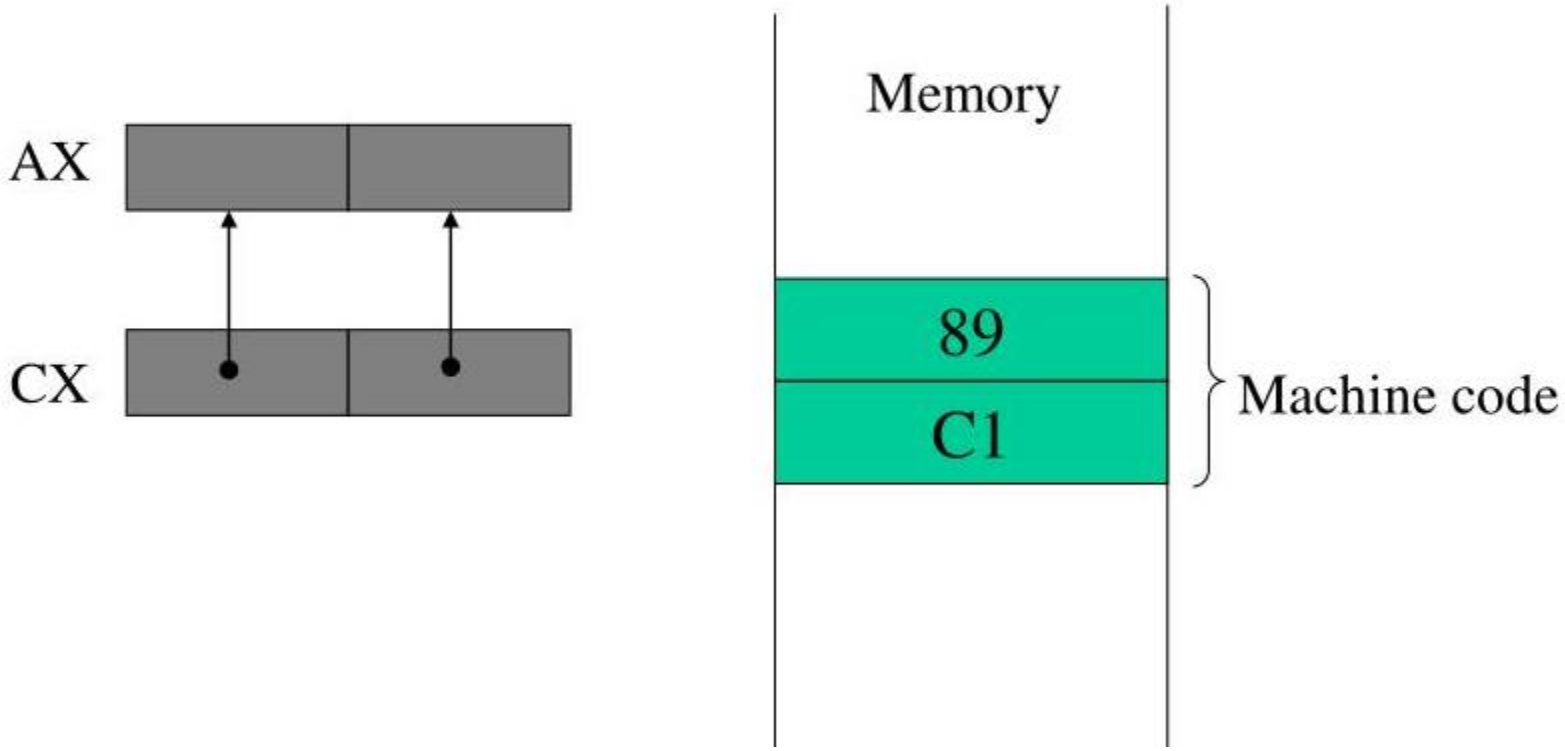
MOV AX,BX



1.4. CHẾ ĐỘ ĐỊA CHỈ CỦA 8086

3. Register addressing mode

Exp : MOV AX, CX

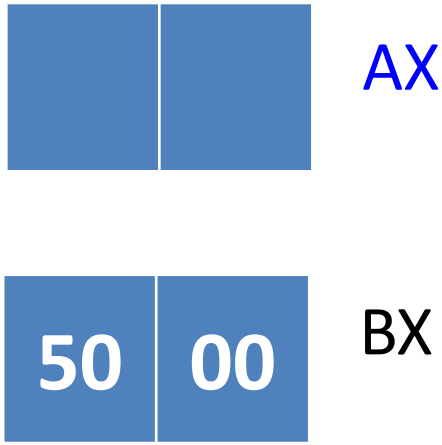


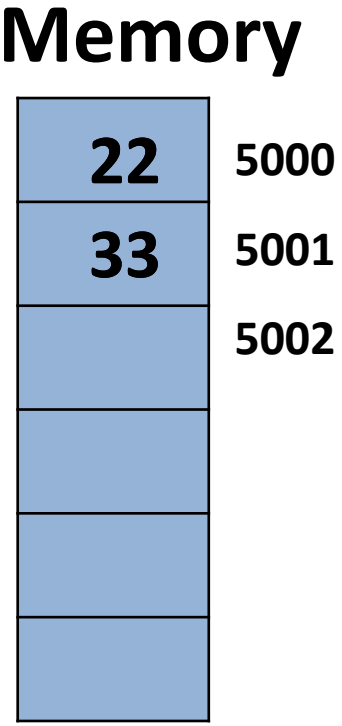
1.4. CHẾ ĐỘ ĐỊA CHỈ CỦA 8086

4. Register Indirect addressing mode

- Địa chỉ của vị trí bộ nhớ chứa dữ liệu hoặc toán hạng được xác định theo cách gián tiếp, sử dụng thanh ghi offset.

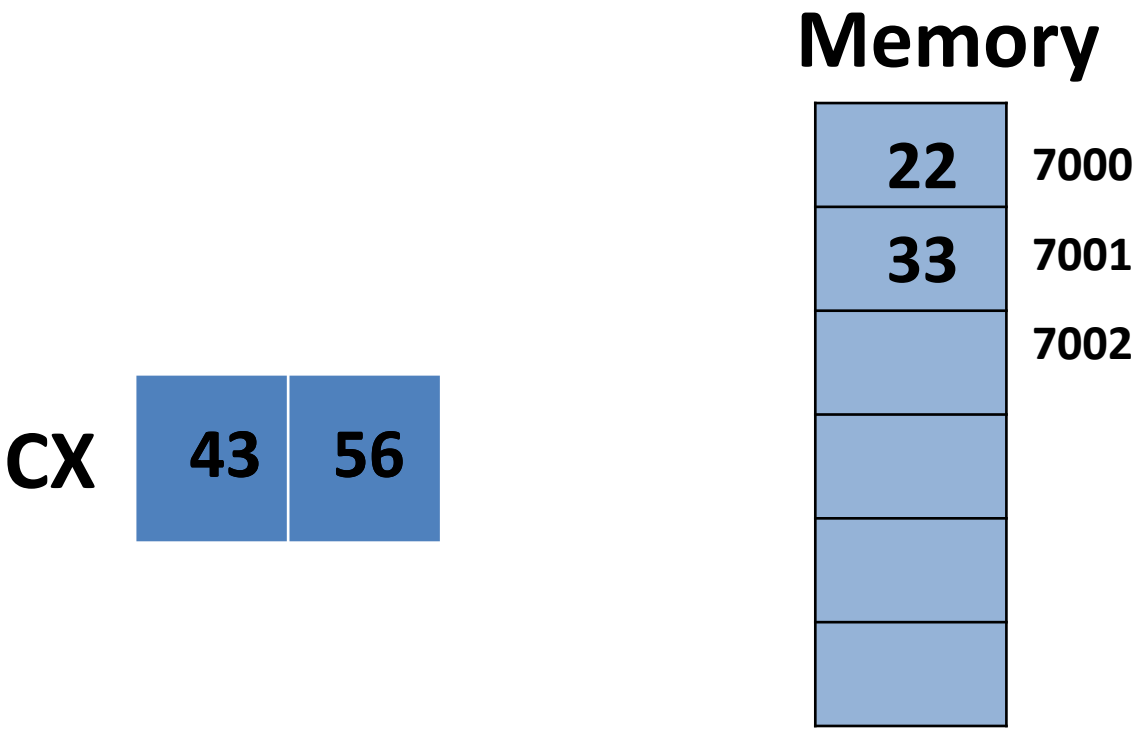
MOV AX,[BX]





4. Register Indirect addressing mode

MOV [7000H],CX

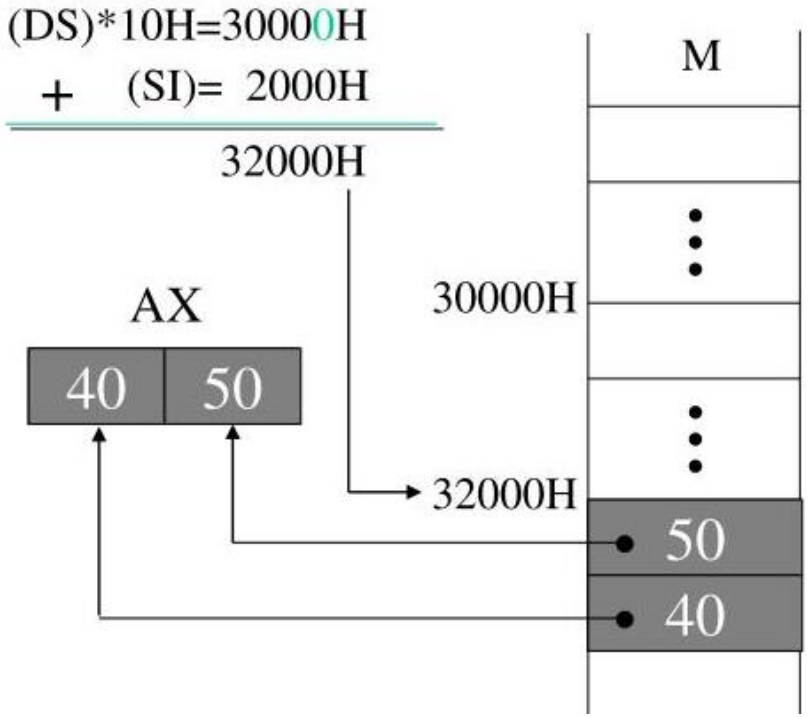


1.4. CHẾ ĐỘ ĐỊA CHỈ CỦA 8086

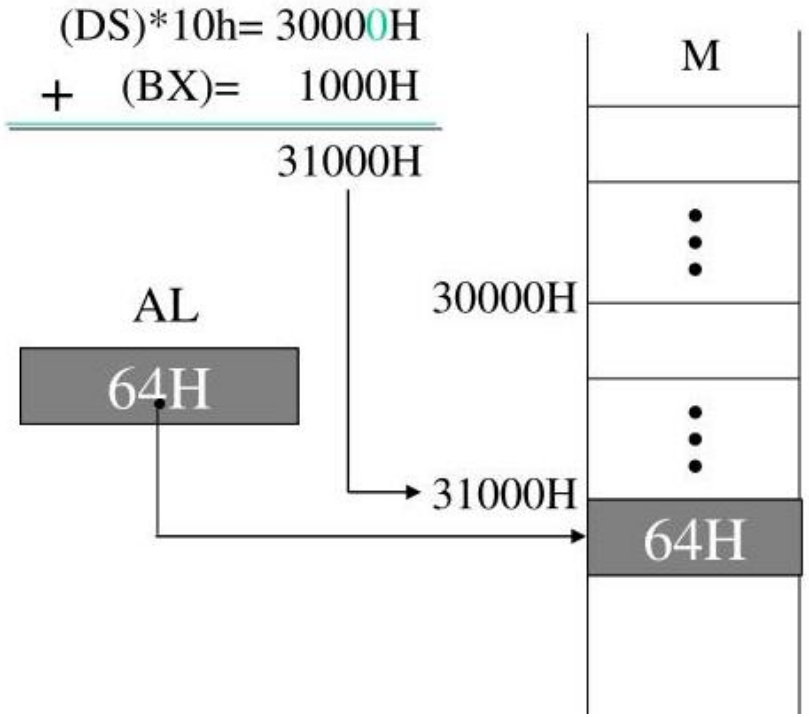
4. Register Indirect addressing mode

ASSUME: (DS)=3000H, (SI)=2000H, (BX)=1000H

MOV AX, [SI]



MOV [BX], AL

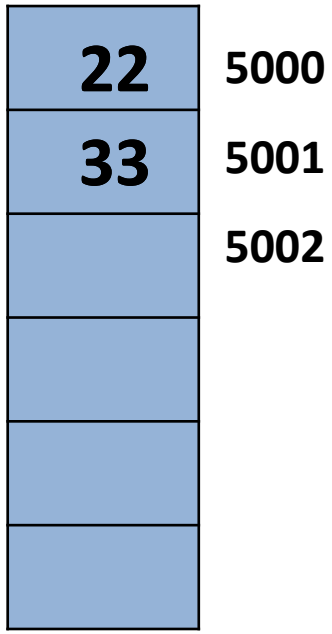


1.4. CHẾ ĐỘ ĐỊA CHỈ CỦA 8086

5. Indexed addressing mode

- Trong chế độ địa chỉ này, offset của toán hạng được lưu trong một trong các thanh ghi chỉ số. DS là thanh ghi đoạn mặc định cho thanh ghi chỉ số SI và DI.

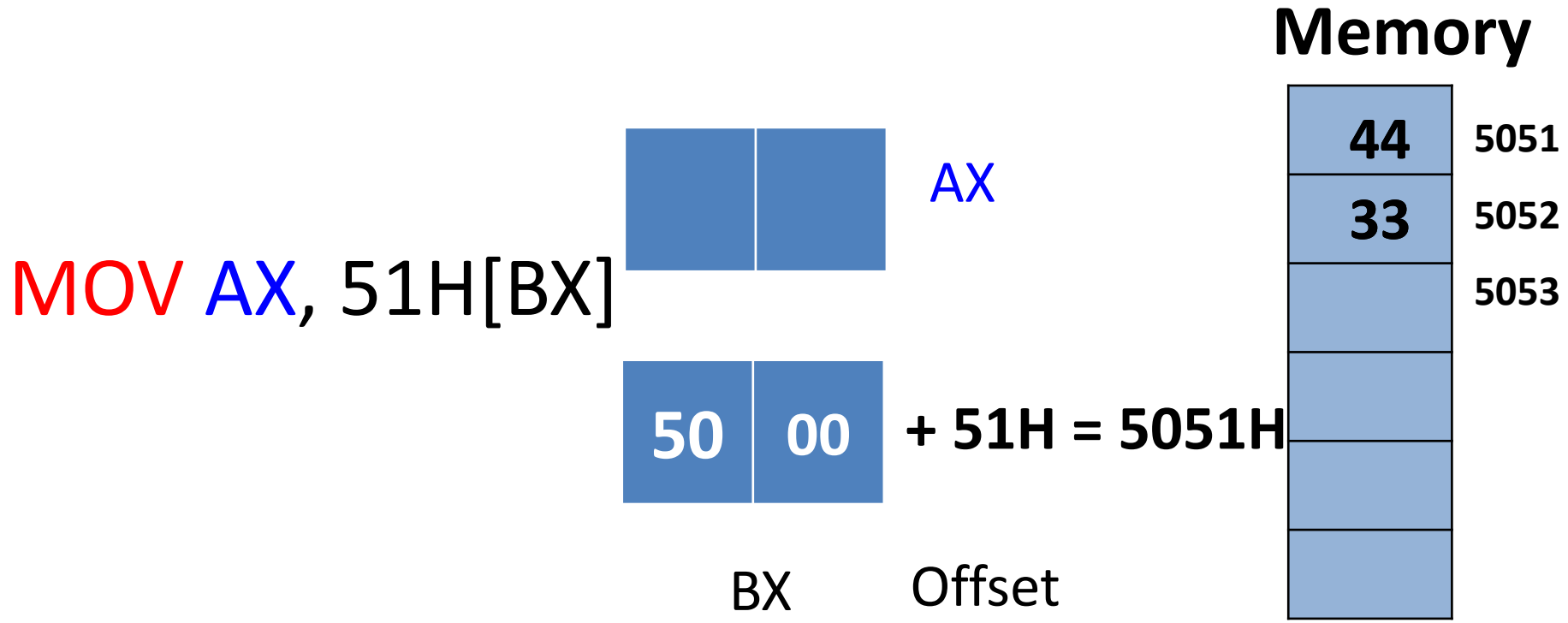
MOV AX,[SI]



1.4. CHẾ ĐỘ ĐỊA CHỈ CỦA 8086

6. Register relative addressing mode

- Trong chế độ này, dữ liệu có sẵn tại một địa chỉ và được xác định bằng cách dịch chuyển thêm 8 bit hoặc 16 bit với nội dung của bất kỳ một trong các thanh ghi BX, BP, SI và DI trong thanh ghi đoạn mặc định (DS hoặc ES) .



1.4. CHẾ ĐỘ ĐỊA CHỈ CỦA 8086

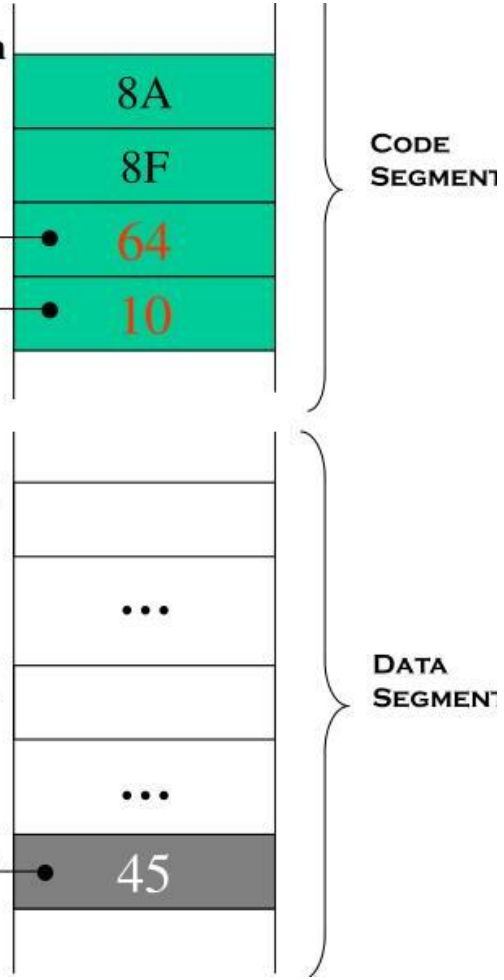
6. Register relative addressing mode

MOV CL, [BX+1064H] ;assume: (DS)=2000h, (bx)=1000h
;Machine Code: 8A8F6410

PA=(DS)*10H+(BX)+1064H

(DS)*10h= 20000H
(BX)= 1000H
\+ 1064H
22064H

CL 45



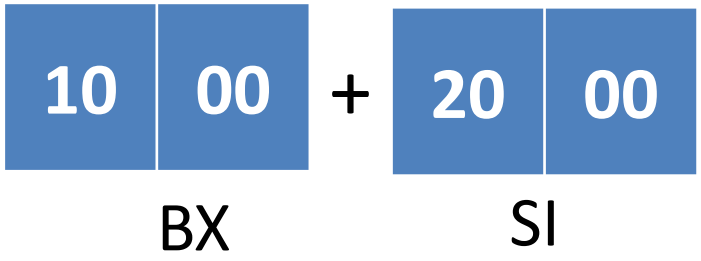
1.4. CHẾ ĐỘ ĐỊA CHỈ CỦA 8086

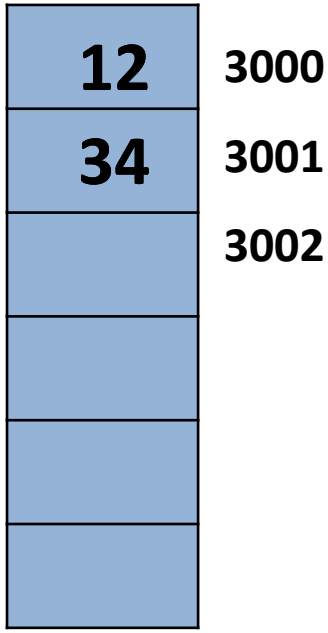
7. Base plus index addressing mode

- Trong chế độ này, địa chỉ được tính bằng cách cộng thêm nội dung của thanh ghi cơ sở (bất kỳ một trong BX hoặc BP) vào nội dung của thanh ghi chỉ mục (SI hoặc DI). Thanh ghi đoạn mặc định là DS.

MOV AX, [BX] [SI]

AX





1.4. CHẾ ĐỘ ĐỊA CHỈ CỦA 8086

7. Base relative plus index addressing mode

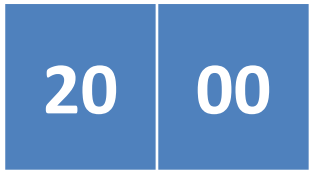
- Địa chỉ được tính bằng cách thêm dịch chuyển 8 hoặc 16 bit với tổng số nội dung của bất kỳ một trong các thanh ghi cơ sở (BX hoặc BP) và bất kỳ một trong các thanh ghi chỉ số, trong một thanh ghi đoạn mặc định.

MOV AX,50H[BX][SI]

AX

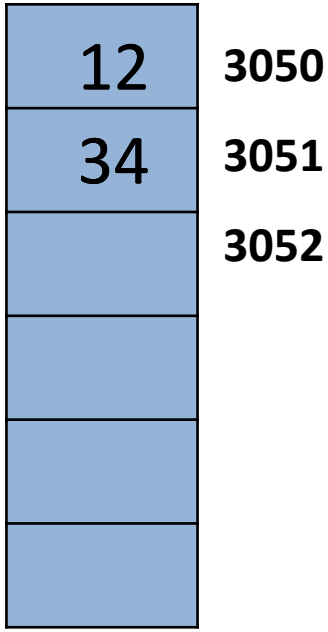
50H + 

BX



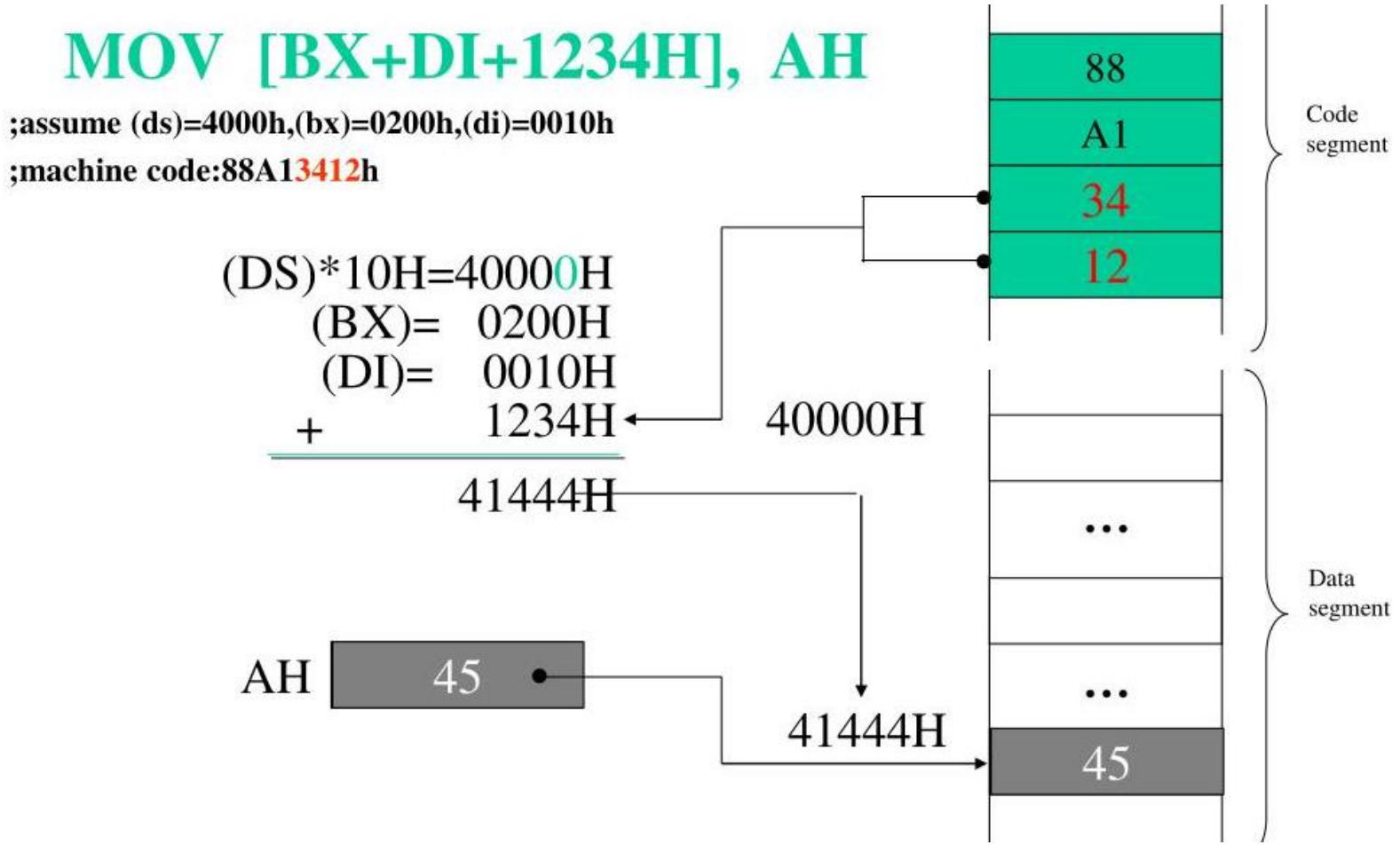
SI

= 3050H



1.4. CHẾ ĐỘ ĐỊA CHỈ CỦA 8086

7. Base relative plus index addressing mode



1.5. TẬP LỆNH

Khái niệm về lệnh và cách mã hóa lệnh

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Mã lệnh	Trợ giúp	Dữ liệu	Dữ liệu	Dữ liệu	Dữ liệu

Định dạng lệnh của 8086

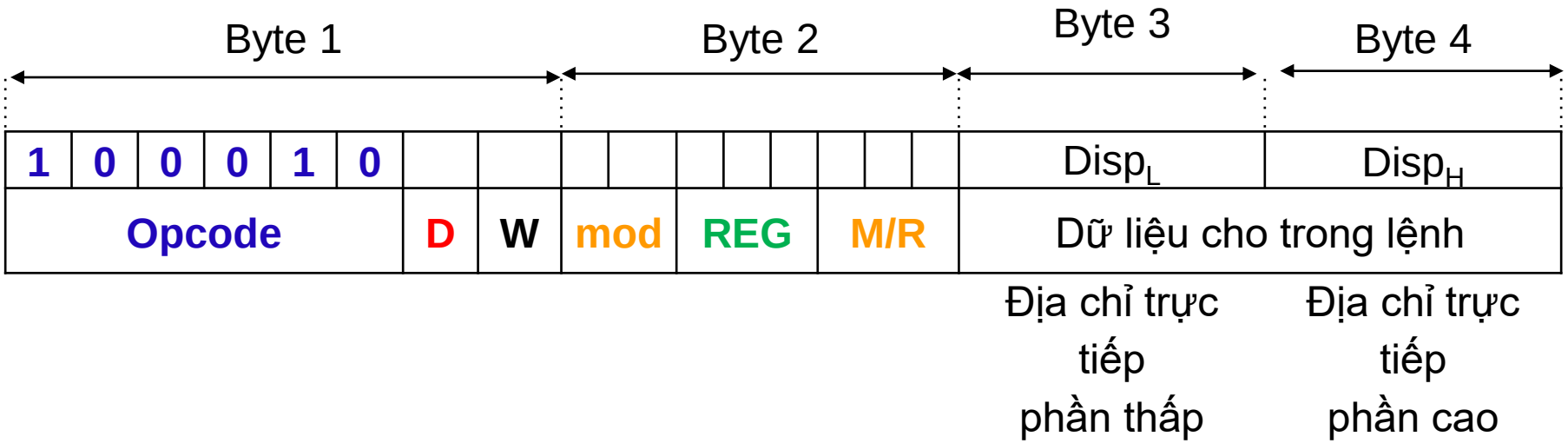
❖**Byte 1:** chứa mã lệnh, xác định thao tác của lệnh. Đây là byte lệnh bắt buộc đối với tất cả các lệnh.

❖**Byte 2:** Nếu lệnh cần phải xác định toán hạng thì có mặt byte 2 để giúp byte 1 xác định toán hạng cho lệnh.

❖**Các byte còn lại byte 3, 4, 5, 6:** để chứa dữ liệu cho lệnh. Tuỳ theo dữ liệu cho trong lệnh mà sẽ xuất hiện các byte khi dịch. Nếu lệnh cần dữ liệu là byte, thì có mặt byte 3. Nếu là Word hay hằng địa chỉ, thì sẽ có thêm byte 3 và 4. Còn nếu cần phải thay đổi hằng địa chỉ cả offset và cả segment, thì cả byte 3, 4, 5, 6 đều có mặt.

1.5. TẬP LỆNH

Khái niệm về lệnh và cách mã hóa lệnh (tiếp)



Dạng thức của lệnh MOV của 8086

1.5. TẬP LỆNH

Khái niệm về lệnh và cách mã hóa lệnh (tiếp)

Thanh ghi		Mã
W=1	W=0	
AX	AL	000
BX	BL	011
CX	CL	001
DX	DL	010
SP	BH	100
DI	AH	111
BP	CH	101
SI	DH	110

Thanh ghi đoạn	Mã
CS	01
DS	11
ES	00
SS	10

1.5. TẬP LỆNH

Khái niệm về lệnh và cách mã hóa lệnh (tiếp)

Phối hợp MOD và R/M để tạo ra các chế độ địa chỉ

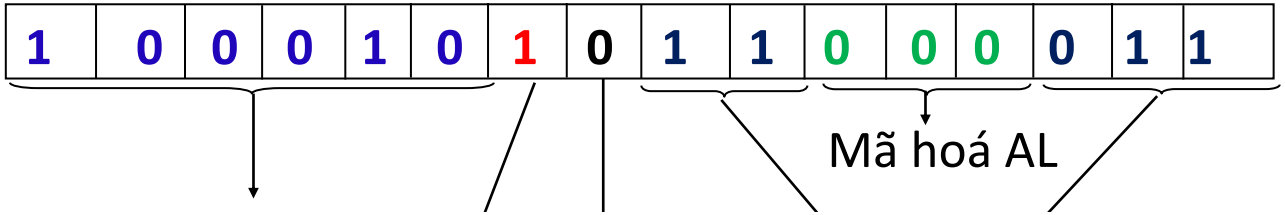
MOD / R/M	00	01	10	11	
				W=0	W=1
000	[BX]+[SI]	[BX]+[SI]+d8	[BX]+[SI]+d16	AL	AX
001	[BX]+[DI]	[BX]+[DI]+d8	[BX]+[DI]+d16	CL	CX
010	[BP]+[SI]	[BP]+[SI]+d8	[BP]+[SI]+d16	DL	DX
011	[BP]+[DI]	[BP]+[DI]+d8	[BP]+[DI]+d16	BL	BX
100	[SI]	[SI]+d8	[SI]+d16	AH	SP
101	[DI]	[DI]+d8	[DI]+d16	CH	BP
110	D16 (địa chỉ trực tiếp)	[BP]+d8	[BP]+d16	DH	SI
111	[BX]	[BX]+d8	[BX]+d16	BH	DI

1.5. TẬP LỆNH

Khái niệm về lệnh và cách mã hóa lệnh (tiếp)

Ví dụ 1: Xét lệnh

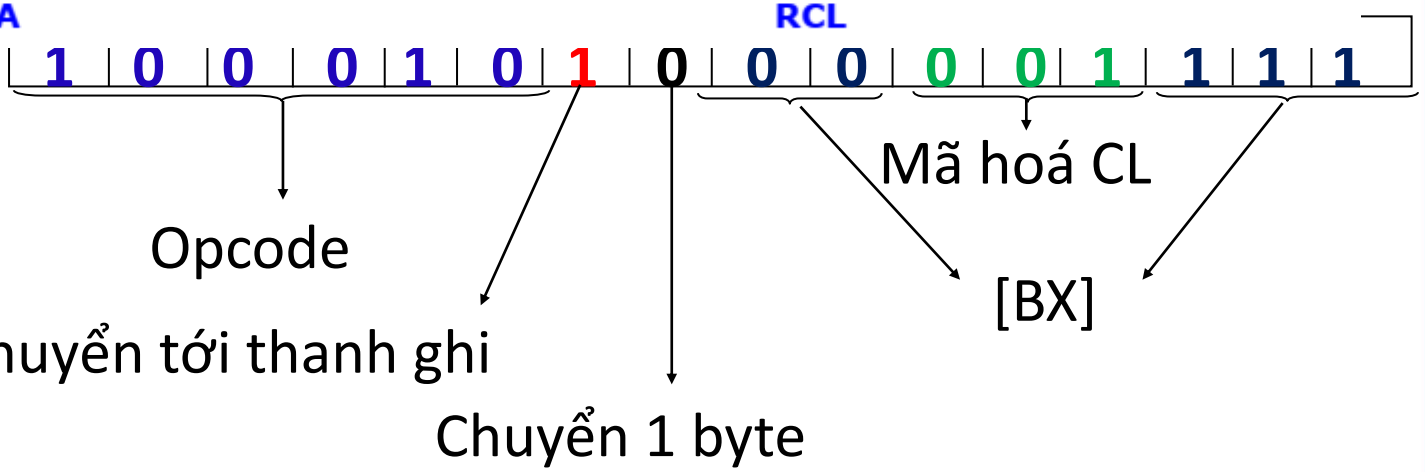
MOV AL, BL



CLI INC JNA JO LOOPNE POPF SAL XCHG
CLI INC JNA JO LOOPNE POPF SAL XCHG
CLI INC JNA JO LOOPNE POPF SAL XCHG
CLI INC JNA JO LOOPNE POPF SAL XCHG
CLI INC JNA JO LOOPNE POPF SAL XCHG
CMC INT JNAE JP LOOPNZ PUSH SAR XLATB
CMP INTO JNB JPE LOOPZ PUSHA SBB XOR
IRET PUSHF
JA RCL

Ví dụ 2: Xét lệnh

MOV CL, [BX]



1.5. TẬP LỆNH

Tập lệnh của 8086

- Các lệnh chuyển dữ liệu
- Các lệnh số học
- Các lệnh logic
- Các lệnh làm việc với xâu ký tự
- Các lệnh nhảy
- Các lệnh khác

(Tham khảo chi tiết trong tài liệu về tập lệnh)

1.5.TẬP LỆNH

Tập lệnh của 8086

AAA
AAD
AAM
AAS
ADC
ADD
AND
CALL
CBW
CLC
CLD
CLI
CMC
CMP
CMPSB
CMPSW
CWD
DAA
DAS
DEC
DIV
HLT
IDIV
IMUL
IN
INC
INT
INTO
IRET
JA
JAE
JB
JBE
JC
JCXZ
JE
JG
JGE
JL
JLE
JMP
JNA
JNAE
JNB
JNBE
JNC
JNE
JNG
JNGE
JNL
JNLE
JNO
JNP
JNS
JNZ
JO
JP
JPE
JPO
JS
JZ
LAHF
LDS
LEA
LES
LODSB
LODSW
LOOP
LOOPE
LOOPNE
LOOPNZ
LOOPZ
MOV
MOVSB
MOVSW
MUL
NEG
NOP
NOT
OR
OUT
POP
POPA
POPF
PUSH
PUSHA
PUSHF
RCL
RCR
REP
REPE
REPNE
REPNZ
REPZ
RET
RETF
ROL
ROR
SAHF
SAL
SAR
SBB
SCASB
SCASW
SHL
SHR
STC
STD
STI
STOSB
STOSW
SUB
TEST
XCHG
XLATB
XOR

2. LẬP TRÌNH HỢP NGỮ

2.1. Cú pháp của một lệnh

Tên Toán tử Toán hạng Chú giải

+ Trường "Tên": được sử dụng để viết nhãn (Label) cho dòng lệnh, cho một biến hoặc cho một thủ tục (procedure). Nhãn là con trỏ tượng trưng để các lệnh tham chiếu tới nó thay cho giá trị địa chỉ của lệnh đó. Chương trình biên dịch sẽ chuyển thành các địa chỉ của bộ nhớ.

Chiều dài từ 1-31 ký tự, là chữ cái, số, các ký tự đặc biệt: ?, @, _, $, %.

Không bắt đầu bằng chữ số và không có dấu cách ở giữa.

Nếu là dấu chấm thì ở đầu.

Trường "Tên" có thể có hay không tuỳ trường hợp cụ thể.

Một nhãn kết thúc bởi dấu (:)

2. LẬP TRÌNH HỢP NGỮ

2.1. Cú pháp của một lệnh

Tên Toán tử Toán hạng Chú giải

+ *Trường "**Toán tử":*** trong trường toán tử nói chung sẽ có các toán tử thật thuộc tập lệnh của vi xử lý và toán tử giả (pseudo-operation).

Toán tử thật biểu thị thao tác của lệnh và sẽ được chương trình dịch ra mã máy.

Toán tử giả không được dịch ra mã máy mà chỉ báo cho chương trình biên dịch làm một việc gì đó.

2. LẬP TRÌNH HỢP NGỮ

2.1. Cú pháp của một lệnh

Tên Toán tử Toán hạng Chú giải

*+ Trường "**Chú giải**":*chứa các lời giải thích làm cho chương trình sáng sủa, dễ hiểu khi viết và khi kiểm tra. Trường chú giải phải bắt đầu bằng dấu (;). Chương trình dịch sẽ bỏ qua khi gặp nó.

***Chú ý*:**

Một câu lệnh của chương trình hợp ngữ bắt đầu từ đầu dòng và kết thúc là xuống dòng

Các trường cách nhau bằng ít nhất bởi dấu cách hoặc dấu tab.

Trường toán tử bắt buộc phải có trong câu lệnh

Có thể dùng một vài dòng để làm chú giải cho một công việc nào đó. Nhưng mỗi dòng chú giải đó phải bắt đầu bằng dấu (;)

2. LẬP TRÌNH HỢP NGỮ

2.2. Dữ liệu cho chương trình

Dữ liệu kiểu số

Ví dụ:

```
0101B          ; số hệ 2
1234           ; số hệ 10
0ABCDh         ; số hệ 16
```

Dữ liệu kiểu ký tự

Được đặt trong cặp dấu trích dẫn đơn hay kép, hoặc dùng trực tiếp mã ASCII

Ví dụ:

```
        "abc"
Hoặc    'abc'
Hoặc    61h, 62h, 63h
```

2. LẬP TRÌNH HỢP NGỮ

2.3. Biến và hằng

Khai báo biến

Toán tử giả	Biểu diễn
DB	Định nghĩa biến kiểu byte
DW	Định nghĩa biến kiểu từ
DD	Định nghĩa biến kiểu từ kép

❖ ***Biến kiểu byte:*** Tên_biến DB giá_trị_khởi_đầu

Ví dụ: B1 DB 4

B2 DB ?

❖ ***Biến kiểu word:*** Tên_biến DW giá_trị_khởi_đầu

Ví dụ: W1 DW 30

2.3. Biến và hằng

Khai báo biến

❖ ***Biến mảng:*** các phần tử cùng loại byte hoặc từ.

Ví dụ: M1 DB 0Ah, 0Bh, 0Ch

Chú ý: *Khi muốn định nghĩa các phần tử có cùng một giá trị khởi đầu thì có thể dùng toán tử DUP như sau*:

Repeat_count DUP(value)

Ví dụ:

M2 DB 10 DUP(0)
M3 DB 10 DUP(?)
M4 DB 4, 5, 6, 7, 8, 9
M5 DB 0,1,2 DUP(3,3 DUP(1),6)

2. LẬP TRÌNH HỢP NGỮ

2.3. Biến và hằng

Khai báo biến

❖***Biến kiểu xâu ký tự :*** Là trường hợp đặc biệt của biến mảng, trong đó các phần tử của mảng là các ký tự.

Ví dụ:

```
S1      DB      'abc'           ; tương đương với
S2      DB      61h, 62h, 63h   ; hoặc
S3      DB      'a', 62h, 63h
```

Khai báo hằng

Tên_hằng EQU hằng_số

Ví dụ:

```
LF      EQU     0Ah
Chao    EQU     'Hello'
```

2. LẬP TRÌNH HỢP NGỮ

2.4. Khung chương trình hợp ngữ (chế độ đơn giản)

❖ **Khai báo quy mô bộ nhớ**

Cú pháp: **.MODEL** kiểu_bộ_nhớ

Kiểu bộ nhớ	Mô tả
TINY	- Cả mã lệnh và dữ liệu nằm trong một đoạn.
SMALL	- Hỗ trợ một đoạn mã lệnh và một đoạn dữ liệu. - Mã lệnh và dữ liệu đều là NEAR
MEDIUM	- Hỗ trợ cho nhiều đoạn mã lệnh, một đoạn dữ liệu. - Mã lệnh là FAR, dữ liệu là NEAR
COMPACT	- Hỗ trợ cho một đoạn mã lệnh, nhiều đoạn dữ liệu. - Mã lệnh là NEAR, dữ liệu là FAR
LARGE	- Hỗ trợ cho nhiều đoạn mã lệnh, nhiều đoạn dữ liệu. - Mã lệnh là FAR, dữ liệu là FAR - Không đoạn nhớ nào lớn hơn 64KB
HUGE	- Hỗ trợ cho nhiều đoạn mã lệnh, nhiều đoạn dữ liệu. - Mã lệnh là FAR, dữ liệu là FAR - Đoạn nhớ có thể lớn hơn 64KB (do người lập trình viết lệnh)

2. LẬP TRÌNH HỢP NGỮ

2.4. Khung chương trình hợp ngữ (trong chế độ đơn giản)

❖ ***Khai báo đoạn ngăn xếp (Stack segment)***

Cú pháp:

.STACK kích_thước

Ví dụ: .STACK 100

❖ ***Khái báo đoạn dữ liệu (Data segment)***

Cú pháp:

.DATA

; Khai báo và gán giá trị cho các biến và định nghĩa hằng

Ví dụ:

```
.DATA
        Chao    DB      'Hello!'
        CR      DB      13
        LF      DB      10
```

❖ ***Khai báo đoạn mã***

Cú pháp:

.CODE tên

2. LẬP TRÌNH HỢP NGỮ

2.4. Khung chương trình hợp ngữ (trong chế độ đơn giản)

❖ *Khai báo đoạn mã*

Cú pháp:

.CODE tên

"Tên": ở đây là tên một đoạn. Khi dùng kiểu SMALL thì không cần phải khai báo tên

```
Tên_CTC    Proc
 ; các lệnh của thân chương trình chính
 CALL      tên_ctc
    :
 Tên_CTC Endp
 ; các lệnh của chương trình con
 RET
 Tên_ctc   Endp
```

2. LẬP TRÌNH HỢP NGỮ

2.4. Khung chương trình hợp ngữ (trong chế độ đơn giản)

❖ *Khung của CT hợp ngữ để dịch ra file có đuôi .EXE*

```
.MODEL    SMALL
.STACK    100
.DATA
          ; các định nghĩa biến và hằng để ở đây
.CODE
          MAIN      PROC
          ; khởi đầu cho DS
                    MOV AX, @data
                    MOV DS, AX
          ; Các lệnh của chương trình chính
          ; Trở về DOS
                    MOV AH, 4Ch
                    INT 21h
          MAIN      ENDP
          ; các chương trình con (nếu có) để ở đây
END MAIN
```

2. LẬP TRÌNH HỢP NGỮ

2.4. Khung chương trình hợp ngữ (trong chế độ đơn giản)

❖ *Khung chương trình hợp ngữ để dịch ra file có đuôi .COM*

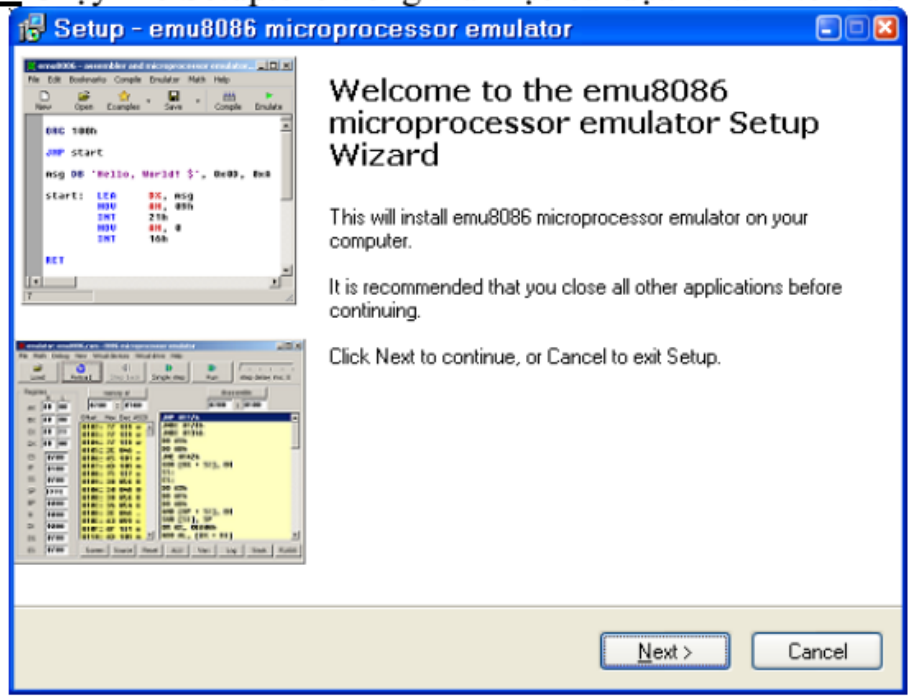
```
.MODEL Small
        .CODE
                ORG     100h
        START:  JMP     CONTINUE
                ; các định nghĩa cho biến và hằng để ở đây
        CONTINUE:
        MAIN    PROC
                ; các lệnh của chương trình chính để ở đây
                INT     20h     ; trở về DOS
        MAIN    ENDP
                ; các chương trình con (nếu có) để ở đây
        END START
```

3. EMU8086

3.1.Cài đặt

Phần mềm này rất đơn giản và dễ cài đặt và có thể dùng free không cần crack!

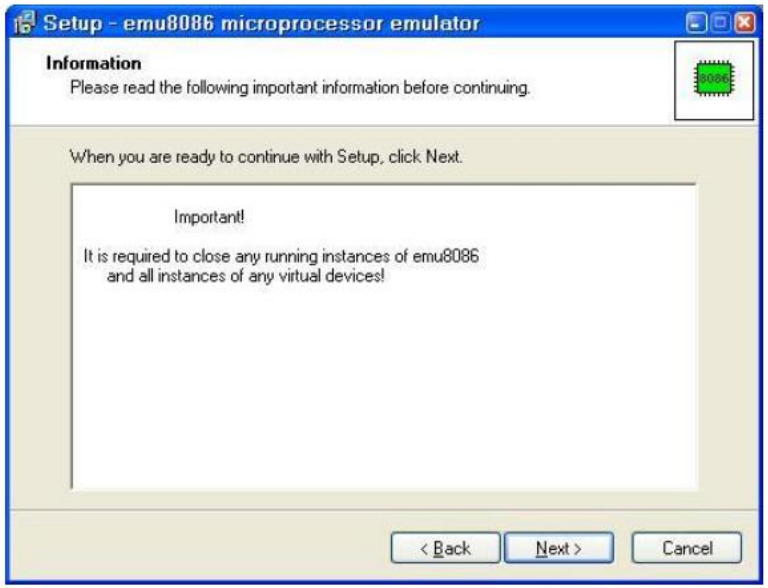
Bước 1: Chạy file Setup.exe trong thư mục cài đặt. Nhấn Next để tiếp tục



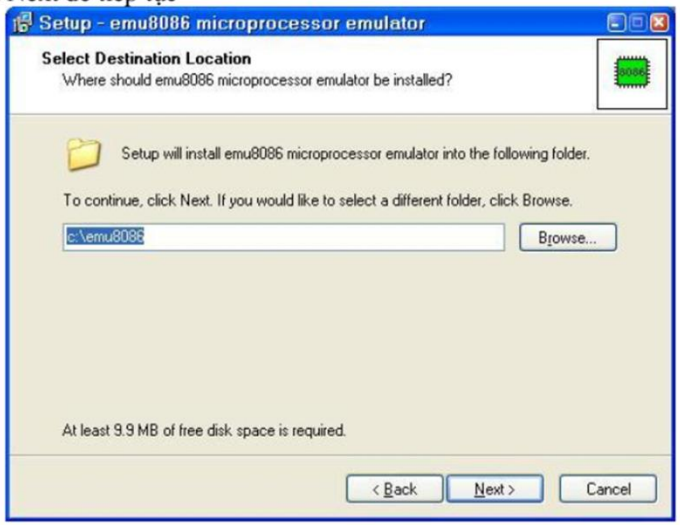
3. EMU8086

3.1.Cài đặt

Bước 2: Trình cài đặt hiện một số thông tin chú ý, nhấn nút Next để bỏ qua



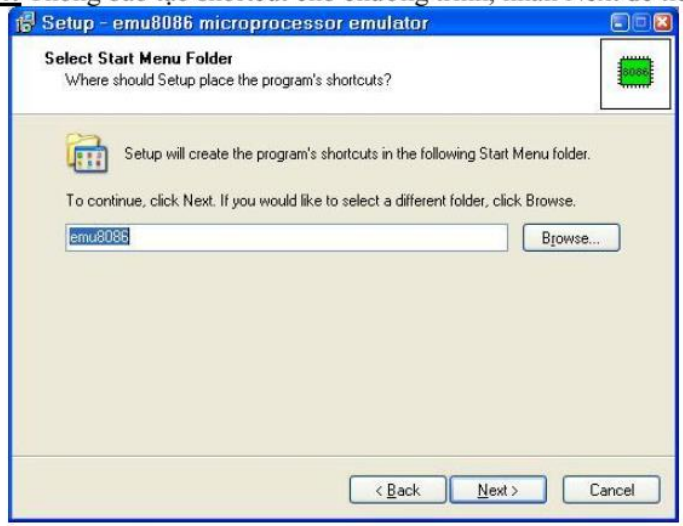
Bước 3: Chọn đường dẫn đến thư mục cài đặt, mặc định là C:\emu8086. Nhấn Next để tiếp tục



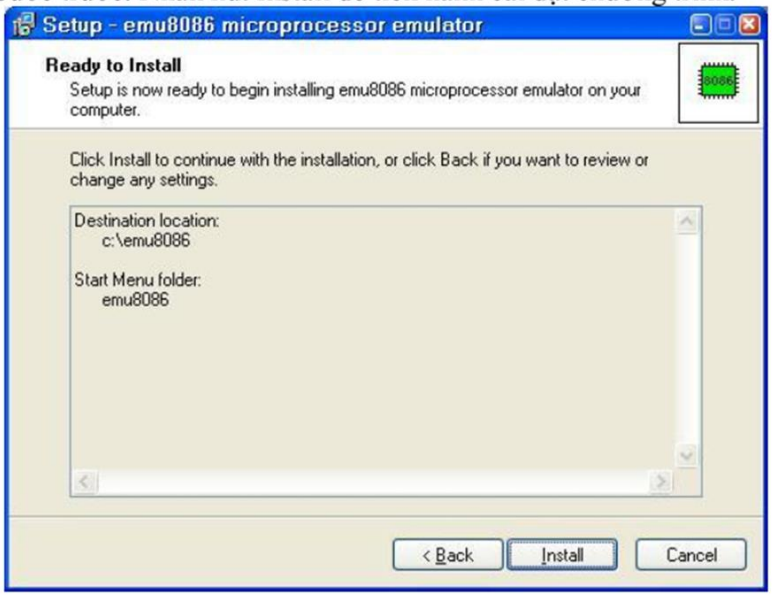
3. EMU8086

3.1.Cài đặt

Bước 4: Thông báo tạo shortcut cho chương trình, nhấn Next để tiếp tục



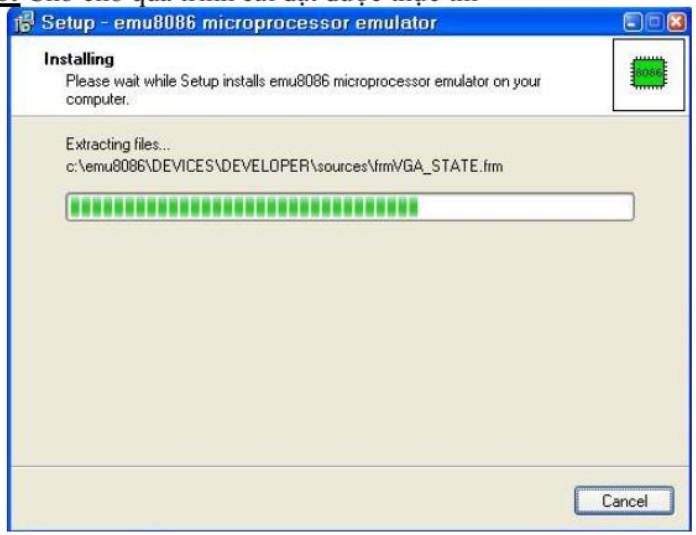
Bước 5: Sau khi trình cài đặt thu thập đủ thông tin, nó hiện thông báo sẵn sàng cho việc cài đặt và tổng hợp các thông tin đã thu thập được trong các bước trước. Nhấn nút Install để tiến hành cài đặt chương trình.



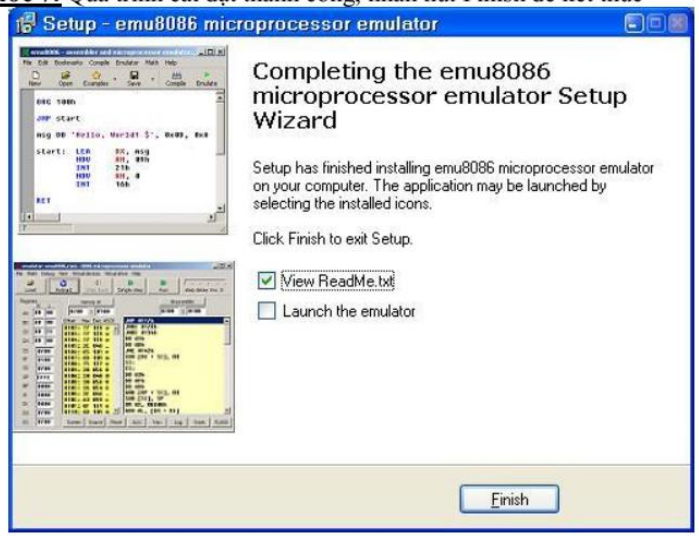
3. EMU8086

3.1.Cài đặt

Bước 6: Chờ cho quá trình cài đặt được thực thi



Bước 7: Quá trình cài đặt thành công, nhấn nút Finish để kết thúc



3. EMU8086

3.1.Cài đặt

Bước 8: Chạy chương trình emu8086 (trên Desktop), giao diện cho lần đầu đăng nhập sẽ hiện ra yêu cầu chúng ta nhập mã đăng ký. Nhấn nút phía dưới (Please Enter the Registration Key) để đăng ký

Bước 9: Nhập thông tin đăng ký (có trong file cd key(8086).txt trong thư mục cài đặt) và nhấn nút OK để kết thúc quá trình đăng ký.

3. EMU8086

3.1.Cài đặt

Bước 8: Chạy chương trình emu8086 (trên Desktop), giao diện cho lần đầu đăng nhập sẽ hiện ra yêu cầu chúng ta nhập mã đăng ký. Nhấn nút phía dưới (Please Enter the Registration Key) để đăng ký

Bước 9: Nhập thông tin đăng ký (có trong file cd key(8086).txt trong thư mục cài đặt) và nhấn nút OK để kết thúc quá trình đăng ký.

3. EMU8086

3.2. Hướng dẫn

Chương trình cho phép chúng ta thực hiện các chức năng chính sau:

-Soạn thảo mã hợp ngữ trên màn hình soạn thảo, dịch ra file .exe hoặc file .com và chạy mô phỏng, debug trực tiếp.

-Tra cứu tập lệnh của bộ vi xử lý 8086 (Help>8086 Instruction Set)

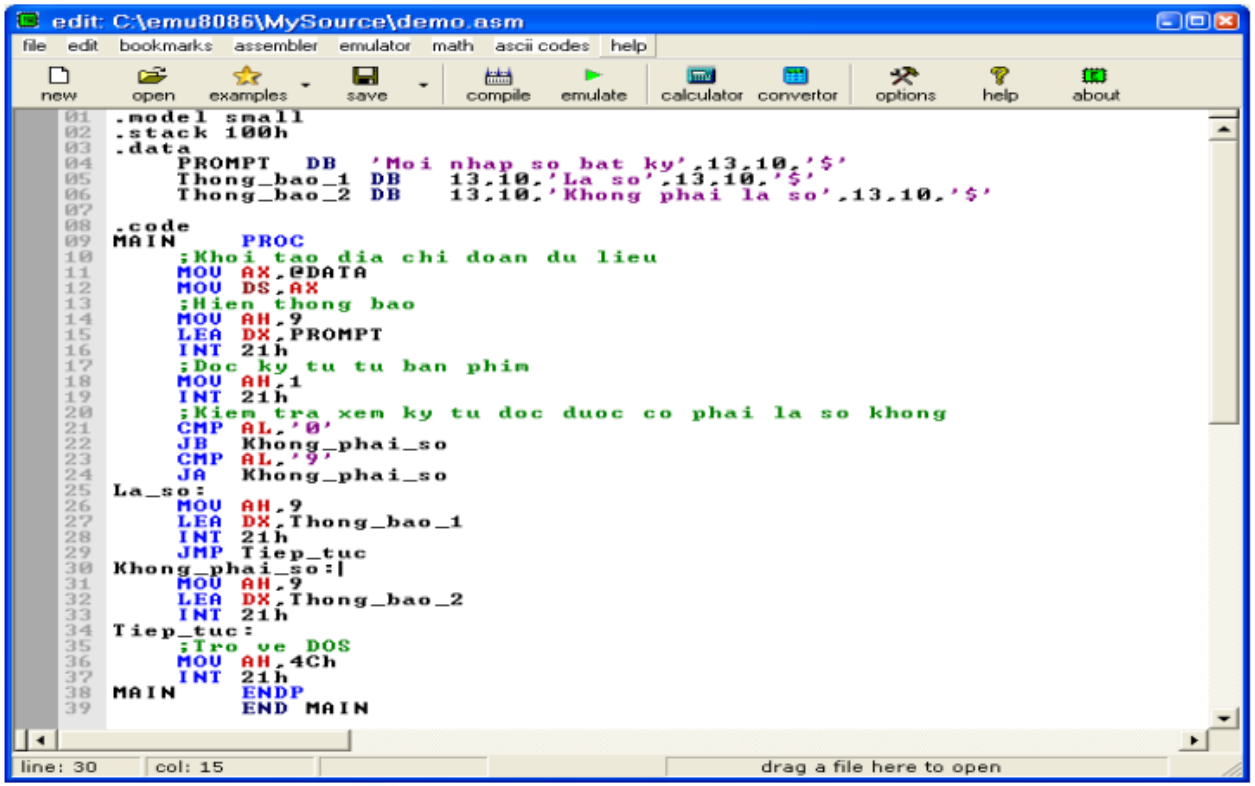
-Tra cứu bảng mã ASCII (Mục ascii codes trên menu)

-Thực hiện các phép toán và chuyển đổi giữa các hệ cơ số thông dụng (nhị phân, thập phân, thập lục phân) (Mục math trên menu)

3. EMU8086

3.2. Hướng dẫn

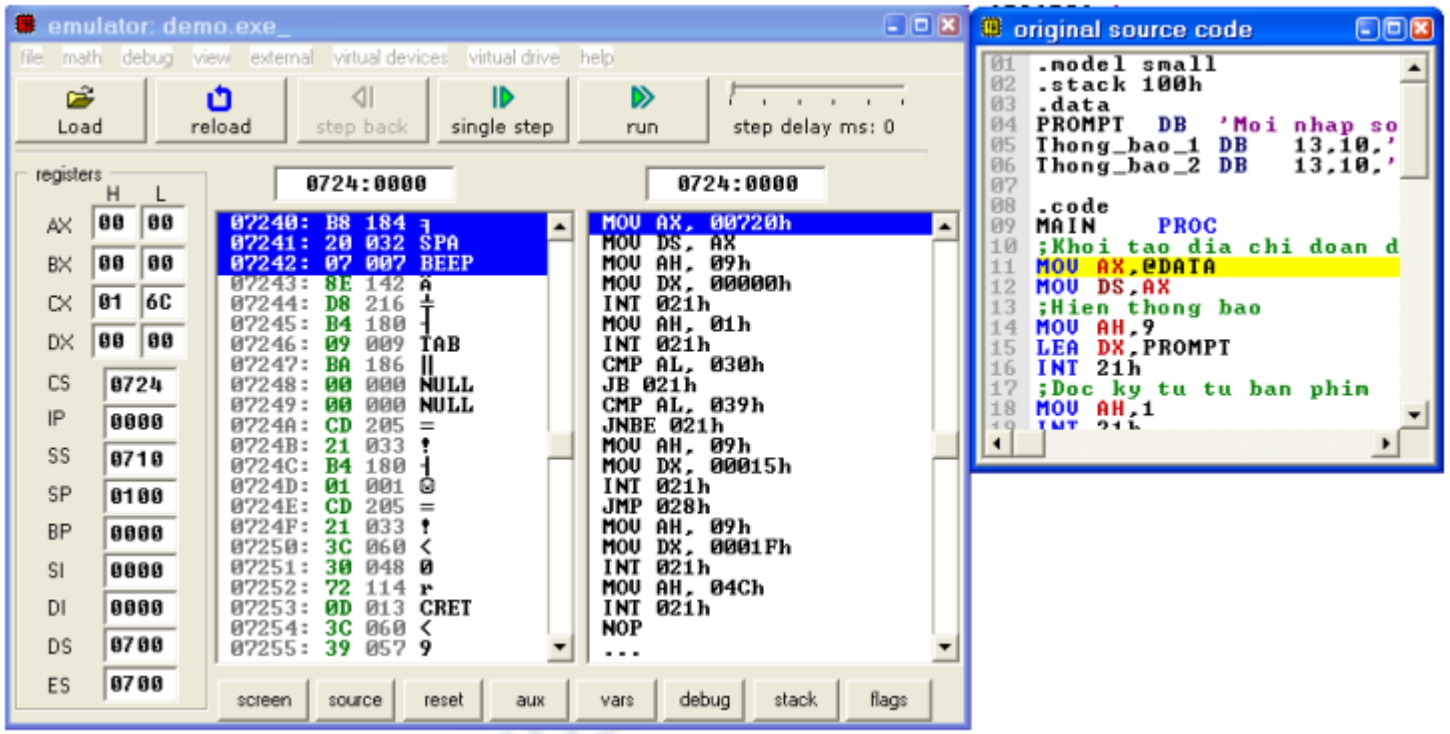
Giao diện:



3. EMU8086

3.2. Hướng dẫn

Giao diện gỡ lỗi



4. THỰC HÀNH

4.1. MỤC TIÊU

- Sử dụng được công cụ Emu8086 để khảo sát các lệnh của Intel-8086.
- Viết đúng cấu trúc của chương trình hợp ngữ dạng .EXE .COM
- Đọc hiểu và sửa lỗi chương trình.

4.2. KIẾN THỨC CẦN CHUẨN BỊ

- Các thao tác cơ bản trên hệ điều hành Windows.
- Cấu trúc chương trình hợp ngữ dạng EXE, COM
- Qui trình Soạn thảo – Dịch chương trình.
- Các lệnh đơn giản của Intel-8086 thường dùng như: MOV, ADD, SUB, INC, DEC, AND, OR. (Tự tra cứu cú pháp)

4. THỰC HÀNH

4.3. NỘI DUNG THỰC HÀNH

4.3.1. Khảo sát lệnh Intel-8086:

a. Nhập vào Emu8086 đoạn lệnh sau đây và dự đoán trước kết quả:

MOV AH, 80 ; AH ← 80 (AX = ?)
MOV AL, 86 ; AL ← 86 (AX = ?)
MOV BX, AX ; BX ← AX (BH = ?, BL = ?)
MOV DH, BL ; DH ← BL (DH = ?, DX = ?)
MOV DL, BH ; DL ← BH (DL = ?, DX = ?)
MOV SI, CS ; SI ← CS (SI = ?)

Thực hiện từng lệnh, sau mỗi lệnh ghi lại kết quả các thanh ghi trong ngoặc để đối chiếu với kết quả dự đoán trên và giải thích.

4. THỰC HÀNH

b. Thực hành tương tự như câu 3.1.1 đối với đoạn lệnh sau:
MOV AX, 8086 ; AX ← 8086 (AH = ?, AL = ?)
ADD AL, 3 ; AL ← AL + 3 (AL = ?, AX = ?)
DEC AX ; AX ← AX – 1 (AH = ?, AL = ?, AX = ?)
SUB AH, 10h ; AH ← AH – 10h (AH = ?, AL = ?, AX = ?)
AND AX, 0FF0h ; AX ← AX and 0FF0h (AX = ?)
c. Sinh viên chủ động lập lại ít nhất 1 lần câu a và b với các giá trị toán hạng khác trong mỗi dòng lệnh.

4. THỰC HÀNH

4.3.2. Chạy chương trình trên Emu8086

a. Chương trình EXE: Điều quan tâm là hiểu cấu trúc chương trình hợp ngữ đã học ở trước

```
DSEG SEGMENT   ; Tạo đoạn DSEG
       chuoi  DB    "Hello World!$"     ; Khai báo biến chuỗi
DSEG ENDS
CSEG SEGMENT   ; Tạo đoạn CSEG
       ASSUME CS: CSEG, DS: DSEG ; CSEG là đoạn lệnh, DSEG là dữ liệu
begin: MOV  AX, DSEG    ; Khởi động địa chỉ đoạn dữ liệu
       MOV  DS, AX
       MOV  AH, 09h     ; AH ← 09h
       LEA  DX, chuoi   ; DX ← địa chỉ offset biến chuoi
       INT  21h         ; gọi ngắt 21h
       MOV  AH, 01h     ; AH ← 01h
       INT  21h         ; gọi ngắt 21h
       MOV  AH, 4Ch     ; Thoát chương trình
       INT  21h
CSEG ENDS
       END  begin
```

4. THỰC HÀNH

4.3.2. Chạy chương trình trên Emu8086

b. Chương trình COM: Điều quan tâm là hiểu cấu trúc chương trình hợp ngữ đã học ở trước

```
org 100h
jmp start
msg:    db      "Hello, World!", 0Dh,0Ah, 24h
start:  mov     dx, msg
        mov     ah, 09h
        int     21h
        mov     ah, 0
        int     16h

ret
```

4. THỰC HÀNH

4. 3.3. Viết các chương trình đơn giản:

[SUM1.ASM] Viết chương trình dạng EXE để tính kết quả biểu thức sau, lưu trữ kết quả trong AX:

10 + 8086 - 100h + 350 + 0FAh

Lưu ý: - Chỉ khai báo 1 đoạn lệnh để viết chương trình.

a. Dịch sửa lỗi *(nếu có lỗi)* và chạy chương trình.

b. Dùng Emu8086 để chạy chương trình trên và kiểm tra kết quả lưu trong AX.

4. THỰC HÀNH

4. 3.3. Viết các chương trình đơn giản:

[SUM2.ASM] Viết chương trình dạng EXE để tính kết quả biểu thức có dạng tổng quát như sau:

KQUA = A + B – C + D + E

Trong đó: KQUA, A, B, C, D, E là các biến 2 byte khai báo trong đoạn dữ liệu.

Lưu ý: - Chương trình gồm 2 đoạn: Đoạn lệnh và Đoạn dữ liệu dùng để chứa các Biến.

a. Gán giá trị các biến A = 1000, B = 10, C = 1Fh, D = 30h, E = 300Ah. Dịch và chạy chương trình.

b. Dùng Emu8086 để kiểm tra kết quả của câu a.

c. Áp dụng SUM2.ASM để tính biểu thức đã cho ở bài Sum1. Dùng Emu8086 để kiểm tra kết quả

4. THỰC HÀNH

4. 3.3. Viết các chương trình đơn giản:

BÀI TẬP

1. Dùng Emu8086 để khảo sát các lệnh khác trong tập lệnh của Intel-8086.

2. Tự tìm hiểu thêm những chức năng khác của Emu8086

3. Viết từng chương trình tính các biểu thức sau: (Phải viết theo kiểu sử dụng biến để chứa toán hạng và kết quả, SV tự đặt tên biến theo ý của mình)

a. 15h * 250

b. 16 * 0AF1h

c. 300 * 400

d. 1000 ÷ 100

e. 1000 ÷ 100h

f. 3AB45Eh ÷ 0A1h