**1. Lý thuyết**

**Câu 1 : Dựa vào công thức MIPS, giải thích tại sao xu thế sản xuất bộ VXL cho máy tính là tăng số nhân ?**

Trả lời :

Công thức MIPS :

Trong đó :

f : Tần số làm việc của bộ VXL

N : Số lượng các đơn vị xử lý số học và logic (ALU) không phụ thuộc vào nhau bên trong bộ vi xử lý

M : Số lượng vi lệnh trung bình của một lệnh trong bộ VXL (thường cần 4-7 vi lệnh)

T: Hệ số thời gian truy nhập bộ nhớ (chu trình chờ đợi trong khi truy nhập bộ nhớ )

Ta thấy

- Với việc tăng f là rất khó khăn bởi f càng tăng thì nhiệt lượng tỏa ra càng lớn, ảnh hưởng tới các thành phần vật lý bên trong, bên cạnh đó, việc tăng tần số khó khăn cũng là do giới hạn về công nghệ, về kiến trúc CPU

- Với việc giảm M, số lượng vi lệnh trung bình của một lệnh thường là 4-7 lệnh, khó thể giảm được hơn nữa

- Với việc giảm T, việc truy nhập bộ nhớ luôn cần một khoảng thời gian nhất định, với giới hạn công nghệ hiện tại, thời gian truy nhập cũng đã ở mức giới hạn

Từ đó, chỉ còn việc tăng N, tức tăng số nhân trên bộ VXL

Cho nên, ta có thể kết luận, xu thế sản xuất bộ VXL cho máy tính là tăng số nhân

**Câu 2 : Phân tích, xác định khi CPU thực hiện lệnh ghi dữ liệu vào bộ nhớ thì các chân ALE và WR có thể cùng tín hiệu tại một thời điểm hay không ?**

Trả lời :

ALE và WR không thể cùng có tín hiệu tại cùng một thời điểm

ALE là xung cho phép chốt địa chỉ. Khi ALE = 1 có nghĩa là đang có tín hiệu địa chỉ

WR là xung cho phép ghi. Khi WR = 1, dữ liệu được phép ghi vào bộ nhớ

Trước khi ghi dữ liệu, phải có tín hiệu của ALE trước để xác định địa chỉ của dữ liệu, sau đó WR mới có tín hiệu cho phép ghi dữ liệu

**Câu 3 : So sánh, phân tích kiến trúc song song và kiến trúc đồng vi xử lý trong vấn đề nâng cao công suất hệ vi xử lý ?**

Trả lời

Kiến trúc song song là kiến trúc kết hợp hai bộ vị xử lý trong khối xử lý trung tâm tạo ra khả năng xử lý song song trong cùng một không gian. Kiểu kiến trúc này có tốc độ xử lý cao gấp đôi so với kiến trúc chỉ dung một bộ vi xử lý

Kiến trúc đồng xử lý sử dụng bộ đồng xử lý, là một bộ vi xử lý riêng biệt kết nối với bộ vi xử lý chính thông qua bus hệ thống. Bộ đồng xử lý chỉ thực hiện một số chức năng đặc biệt, ví dụ như các phép toán đòi hỏi sự chính xác sử dụng dấu phẩy động. Tốc độ của bộ đồng xử lý ở những phép tính này sẽ nhanh hơn rất nhiều so với bộ xử lý chính

Có thể thấy, với kiến trúc song song, việc kết hợp song song hai bộ vi xử lý tạo ra tốc độ xử lý tổng thế nhanh gấp đôi kiến trúc đơn vi xử lý

Còn với kiến trúc đồng xử lý, bộ đồng xử lý cho tốc độ xử lý cao hơn bộ xử lý chính ở các tác vụ nhất định

**Câu 4 : Phân tích, xác định khi CPU thực hiện việc chấp nhận yêu cầu trao đổi dữ liệu kiểu DMA thì ngay sau đó CPU có thể điều khiển bộ nhớ truyền dữ liệu vào trong CPU được không ?**

Trả lời :

KHÔNG ! Trước khi CPU có thể điều khiển bộ nhớ truyền dữ liệu vào trong CPU, CPU phải đợi tín hiệu yêu cầu treo CPU (HOLD[I]) để mạch ngoài thực hiện việc trao đổi dữ liệu với bộ nhớ bằng cách thâm nhập trực tiếp. Khi HOLD =1, CPU 8088 sẽ tự tách ra hệ thống bằng cách treo tất cả các bus A, bus D, bus C của nó để bộ điều khiển DMA có thể lấy được quyền điều khiển hệ thống để làm công việc trao đổi dữ liệu

**Câu 5 : Phân tích cấu trúc bộ VXL 8088,xác định cơ chế thực hiện một lệnh của 8088 ?**

Trả lời :

\* Cấu trúc bộ VXL 8088 :

- Bao gồm hai khối chính : Khối phối ghép (BUS INTERFACE UNIT – BIU) và khối thực hiện lệnh (EXECUTION UNIT)

- BIU đưa ra địa chỉ, đọc mã lệnh từ bộ nhớ, đọc/ghi dữ liệu từ vào cổng hoặc bộ nhớ. Nói cách khác BIU chịu trách nhiệm đưa địa chỉ ra bus và trao đổi dữ liệu với bus

- Trong EU bao gồm khối điều khiển (CU) để điều khiển các hoạt động bên trong và bên ngoài CPU, và khối số học logic (ALU) dùng để thực hiện các thao tác khác nhau với các toán hạng của lệnh. Tóm lại, khi CPU hoạt động, EU sẽ cung cấp thông tin về địa chỉ cho BIU để khối này đọc lệnh và dữ liệu, còn bản thân nó thì đọc lệnh và giải mã lệnh

- Ngoài ra còn có các thanh ghi đoạn, thanh ghi đa năng, thanh ghi cờ

\* Cơ chế thực hiện một lệnh :

- Bao gồm 3 giai đoạn : Đọc mã lệnh (OPCODE FETCH), giải mã lệnh (DECODE), thực hiện lệnh (EXECUTION)

- Việc đọc mã lệnh do khối BIU thực hiện, giải mã và thực hiện lệnh do khối EU đảm nhiệm. Các khối chức năng này có khả năng làm việc đồng thời và các bus liên tục được sử dụng

**Câu 6 : Phân tích, xác định khi chân INTR của 8088 có tín hiệu thì các chân AD0 – AD7 có thể có tín hiệu hay không, và đó là tín hiệu gì ?**

Trả lời :

- Có tín hiệu, đó là tín hiệu dữ liệu về số hiệu của ngắt.

- Khi chân INTR có tín hiệu có nghĩa là có yêu cầu ngắt -> bộ vi xử lý có thể chấp nhận hoặc không chấp nhận xử lý ngắt. Nếu chấp nhận thì có cảnh báo thông qua chân INTA.

- CPU không biết là ngắt hay không, nếu chấp nhận thì báo INTA là ngắt -> đưa số hiệu ngắt vào-đưa dữ liệu vào thông qua chân AD0-AD7

- Khi INTR có tín hiệu-tín hiệu số hiệu dữ liệu của ngắt,nếu CPU đồng ý thì báo thông qua chân INTA -> gửi số hiệu ngắt vào trong bộ vi xử lý thông qua các chân AD0-AD7-tín hiệu dữ liệu về số hiệu của ngắt.

**Câu 7 : Phân tích muốn kiểm tra điều kiện biến Var1 có nhỏ hơn biến Var2 hay không thì bộ VXL 8088 sẽ phải thực hiện những thao tác gì ?**

Trả lời :

Để kiểm tra điều kiện biến Var1 có nhỏ hơn(lớn hơn hoặc bằng) biến Var2 hay không, bộ vi xử lý sẽ tiến hành thực hiện phép trừ ảo và dựa vào giá trị của các cờ để đưa ra kết quả so sánh

Cụ thể, bộ vi xử lý sẽ thực hiện phép trừ ảo Var1- Var2

Nếu ZF = 1, tức Var1 – Var2 = 0, thì Var 1 = Var 2

Nếu ZF = 0, SF = 1 (kết quả âm), tức Var 1 – Var2 < 0, thì Var1<Var2

Nếu ZF = 0, SF = 0(kết quả dương), tức Var1-Var2>0, thì Va1>Var2

**Câu 8 : Phân tích, xác định trong trường hợp tín hiệu từ chân AD3 của 8088 đi đến Bus dữ liệu thì có những chân tín hiệu nào liên quan ?**

Trả lời :

- Khi tín hiệu từ chân AD3 đi đến bus dữ liệu thì truyền tín hiệu dữ liệu, để truyền được tín hiệu dữ liệu thì chân DEN=1 và ALE=0

- DT/R xác định chiều, chiều đi hướng từ trong ra ngoài

- Tín hiệu có thể đi đến bộ nhớ hoặc thiết bị ngoại vi, do IO/M quyết định.

**Câu 9 : Phân tích ý nghĩa, cơ chế làm việc của việc dồn kênh và phân kênh trong hệ VXL  
Trả lời :**

Để giảm bớt khó khăn về mặt công nghệ do việc phải chế tạo nhiều chân cho các tín hiệu của vi mạch CPU, người ta đã tìm cách hạn chế số chân của vi mạch bằng cách dồn kênh nhiều tín hiệu trên cùng một chân ( ví dụ, dồn tín hiệu địa chỉ với tín hiệu dữ liệu vào cùng chân AD để đưa ra ngoài)

Khi nhận được các tín hiệu dồn kênh, ta tiến hành phân kênh, nhằm tách các tín hiệu để tái tạo lại tín hiệu gốc cho các bus độc lập (bus địa chỉ và bus dữ liệu). Việc này thực hiện bằng cách sử dụng các vi mạch có chức năng thích hợp ở bên ngoài (thường là mạch chốt). Để tách thông tin, CPU đưa ra thêm xung ALE để báo cho bên ngoài biết tín hiệu lúc này là địa chỉ hay dữ liệu, xung ALE lúc này được dùng để mở các mạch chốt và tách được các thông tin về địa chỉ bị dồn kênh

Cơ chế tương tự với các chân dồn địa chỉ / trạng thái

**Câu 10 : Phân tích, xác định có trường hợp nào cả hai chân DEN và chân IO/M của 8088 cùng có tín hiệu hay không ?**

Trả lời :

Có trường hợp cả hai chân DEN và IO/M cùng có tín hiệu

Chân IO/M có tín hiệu phân biệt trong thời điểm đã định phân tử nào trong các thiết bị vào ra hoặc bộ nhớ được làm việc với CPU, còn chân DEN có tín hiệu báo cho bên ngoài biết là lúc này trên bus dồn kênh AD có dữ liệu ổn định

Khi chân DEN có tín hiệu báo có dữ liệu ổn định trên bus dồn kênh AD, cùng lúc đó, chân IO/M có tín hiệu cho phép bộ nhớ làm việc với CPU

**Câu 11 : Phân tích xác định các thành phần chính cần phải có trong một bộ VXL ?**

Trả lời :

Có ba thành phần chính cần phải có trong một bộ VXL

1. ALU : Tập hơp vi mạch logic, là đơn vị thực hiện các phép toán, phép logic, phép so sánh …

2. CU : Đơn vị logic điều khiển, chỉ bảo bộ vi xử lý giải mã và thực hiện chương trình như thế nào

3. Thanh ghi (Register) : Lưu trữ dữ liệu

**Câu 12 : Phân tích xác định khi chân TEST có tín hiệu thì chân ALE có thể có tín hiệu hay không ?**

- Chân TEST có tín hiệu dùng để xác định khi nào kết thúc lệnh WAIT.

- Bộ VXL phải thực hiện lệnh WAIT cuối chu kỳ T3, chân TEST dùng để xác định lệnh WAIT. Tại chu kỳ T3 đã trải qua tiến trình truyền dữ liệu -> ALE=0. Vì vậy TEST và ALE không thể có cùng tín hiệu.

**Câu 13 : Phân tích, xác định thông số T trong công thức MIPS có liên quan thế nào đến bộ nhớ Cache ?**

Trả lời :

Trong công thức MIPS, thông số T là hệ số thời gian truy cập bộ nhớ, khi thông số T càng nhỏ thì tốc độ của bộ vi xử lý sẽ càng tăng

Bộ nhớ cache chính là tác nhân làm giảm thông số T này

Cache là bộ nhớ có tốc độ cao, được thiết kế đề giảm thời gian truy nhập bộ nhớ cho bộ vi xử lý. Cache có thể nằm ngay bên trong bộ vi xử lý hoặc nằm bên cạnh bộ vi xử lý và được kết nối trực tiếp với bộ vi xử lý, chính nhờ sự kết nối trực tiếp đó, làm cho thời gian truy xuất bộ nhớ cache nhanh hơn rất nhiều so với việc CPU phải truy xuất vào bộ nhớ chính (được kết nối với CPU qua bus hệ thống)

Vì vậy, nhờ có bộ nhớ cache, thông số T giảm đi, tốc độ của bộ vi xử lý tăng lên

**Câu 14 : Phân tích, xác định khi CPU thực hiện thao tác đọc dữ liệu từ bộ nhớ, có phải chân tín hiệu Ready sẽ nhận tín hiệu trước khi chân DT/R truyền tín hiệu ra ngoài hay không** ?

Trả lời :

Khi CPU đọc dữ liệu từ bộ nhớ, muốn có tín hiệu READY thì phải biết READY xuất phát từ đâu, vì vậy phải đưa ra tín hiệu địa chỉ có nghĩa là tín hiệu đưa từ trong ra ngoài do DT/R quyết định, vì vậy DT/R phải có tín hiệu để xác định chiều chuyển từ trong ra ngoài tại thời điểm truyền địa chỉ. Sau khi truyền địa chỉ xong, READY mới có tín hiệu.

**Câu 15 : Phân tích, xác định tính chất gì trong việc xác định công suất bộ xử lý ảnh hưởng đến khả năng tăng dung lượng bộ nhớ trong hệ vi xử lý**

Trả lời :

Khả năng đánh địa chỉ là một tính chất trong việc xác định công suất bộ vi xử lý ảnh hưởng đến khả năng tăng dung lượng bộ nhớ trong hệ vi xử lý. Các từ dữ liệu và lệnh máy cất trong bộ nhớ tại các ngăn nhớ khác nhau. Mỗi ngăn nhớ phải có một địa chỉ để nhận biết. Khi dải đánh địa chỉ càng lớn thì dung lượng nhớ sẽ càng nhiều

**Câu 16 : Phân tích, xác định khi trên bus dữ liệu đang có dữ liệu được đưa vào CPU, thì những chân tín hiệu nào của CPU cũng đang có tín hiệu ?**

Trả lời :

- Khi trên bus dữ liệu đang có dữ liệu được đưa vào CPU có nghĩa là bus dữ liệu đang đưa dữ liệu vào CPU qua chân AD0-AD7, muốn AD0-AD7 truyền tín hiệu dữ liệu thì DEN có tín hiệu, chiều đi vào do DT/R quyết định.

- Khi có dữ liệu vào thì lệnh đọc dữ liệu được thực hiện-> chân RD có tín hiệu.

**2. Bài tập**

**2.1. DẠNG 1 – LẬP TRÌNH ASM**

***2.1.1. Các câu lệnh cơ bản xuất hiện trong đề cương***

**- MOV A, B :** Chuyển giá trị lưu trong thanh ghi B (giá trị B) vào thanh ghi A

VD : MOV AX, A304H : Chuyển giá trị A304 lưu vào trong thanh ghi AX

MOV AX,BX : Chuyển giá trị lưu trong thanh ghi BX vào thanh ghi AX

**- SHL A,B :** Dịch trái thanh ghi A đi B bit (B lần)

VD : SHL AH,4 : Dịch trái thanh ghi AH đi 4 bit

Cụ thể, giả dụ thanh ghi AH có giá trị là FF, đổi ra nhị phân sẽ là :

1111 1111

Dịch trái đi 4 bit là hành động : đẩy 4 bit đầu tiên của thanh ghi AH sang trái và chèn them 4 bit 0 vào cuối, lúc này giá trị sẽ là : 1111 0000 (F0)

**- SHR A,B :** Dịch phải thanh ghi A đi B bit (B lần)

(Cách thức hoạt động ngược lại với dịch trái)

**- ROL A,B :** Quay trái A đi B bit

VD : ROL AH,4 : Quay trái thanh ghi AH đi 4 bit

Cụ thể, giá dụ thanh ghi AH có giá trị là AB, đổi ra nhị phân sẽ là :

1010 1011

Quay trái AH đi 4 bit là hành động : Nhấc lần lượt 4 bit bên trái nhất (1010) chuyển xuống vị trí cuối cùng, lúc này giá trị sẽ là : 1011 1010

**- ROR A,B :** Quay phải A đi B bit

(Cách thức hoạt động ngược lại với quay trái)

**- CMP A,B :** So sánh A với B

**- INC A :** Tăng A lên 1 đơn vị

**- DEC A :** Giảm A đi 1 đơn vị

**- ADD A,B :** Lấy A+B, kết quả lưu vào A

**- SUB A,B :** Lấy A-B, kết quả lưu vào A

**- DIV BL :** Lấy giá trị của thanh ghi AL chia cho giá trị của thanh ghi BL, thương lưu vào AL, dư lưu vào AH

**- MUL BL :** Lấy giá trị của thanh ghi AL nhân với giá trị của thanh ghi BL, tích lưu vào AX

**- LOOP :** Lệnh lặp, lấy giá trị của thanh ghi CX (hoặc CL) làm số lần lặp, mỗi lần lặp, thanh ghi này giảm đi 1 đơn vị

**XOR A,B :** Phép “XOR” A với B

Một chút căn bản :

1 xor 1 = 0

0 xor 0 = 0

0 xor 1 = 1

1 xor 0 = 1

Trong đề cương, có một bài, trong đó có 1 câu lệnh là:

**XOR AX,AX ;** Câu lệnh này có tác dụng reset giá trị thanh ghi AX về 0

**\* LỆNH NHẢY :**

Thông thường, đằng trước các câu lệnh nhảy, sẽ có một câu lệnh so sánh dạng CMP A,B. Các lệnh nhảy sắp liệt kê dưới đây sẽ dựa vào kết quả so sánh của câu lệnh này để quyết định xem có nhảy hay không

**JNE** (Jump If Not Equal) : Nhảy nếu không xảy ra dấu “ = ”

**JA** (Jump If Above) : Nhảy nếu lớn hơn

**JB** (Jump If Below) : Nhảy nếu nhỏ hơn

**JNZ** (Jump If Not Zero) : Nhảy nếu không bằng “0”

***2.1.1. Ví dụ cụ thể***

Câu 5 : Xác định giá trị AX sau khi thực hiện đoạn mã sau

MOV AX,0055H ; Lưu giá trị 0055H vào AX

MOV BL,02H ; Lưu giá trị 02H vào BL

MOV CX,0003H ; Lưu giá trị 0003H vào CX

LAP1:

DIV BL ; Lấy AL chia BL – Thương lưu AL, dư lưu AH

XOR AH,AH ; Reset AH về 0

LOOP LAP1; Lặp nhãn “LAP1”, mỗi lần Loop, CL giảm đi một đơn vị

CL = 3 -> Lặp 3 lần

KẾT QUẢ : AX = 000AH

**Note : Để kiểm tra kết quả, copy đoạn mã vào trong Emu8086 rồi chạy**

**2.2. DẠNG 2 – GIẢI MÃ LỆNH, TÍNH TOÁN ĐỊA CHỈ LOGIC – ĐỊA CHỈ VẬT LÝ**

**Câu 1 :** Cho các giá trị DX = 923Ah, BX = 83ADh, DS = 9F2Ch, [A76CAh] = E6h. Xác định ô nhớ có địa chỉ vật lý B5B9Eh lưu giá trị bao nhiêu sau khi CPU thực hiện ba lệnh liên tiếp sau đây : 8A775Dh, 8BDAh, 88B7A482h

Giải :

**Bước 1 : Giải mã lệnh**

**1.1. 8A775Dh**

+ Đổi mã hexa ra mã nhị phân :

1000 1010 0111 0111 0101 1101

+ Tiến hành tách dãy nhị phân theo thứ tự : **6 1 1 2 3 3**

**100010**  **1**  **0** **01**  **110** **111** 5D

**OPCODE**  **D** **W** **MOD** **REG** **R/M**

Câu lệnh sẽ có dạng : **OPCODE ĐÍCH, NGUỒN**

( Chỉ cần đổi 4 kí tự hexa đầu tiên, các kí tự sau đó không cần thiết phải đổi, bởi những kí tự đó là phàn địa chỉ dịch, chút nữa sẽ ghi trực tiếp vào trong câu lệnh giải mã )

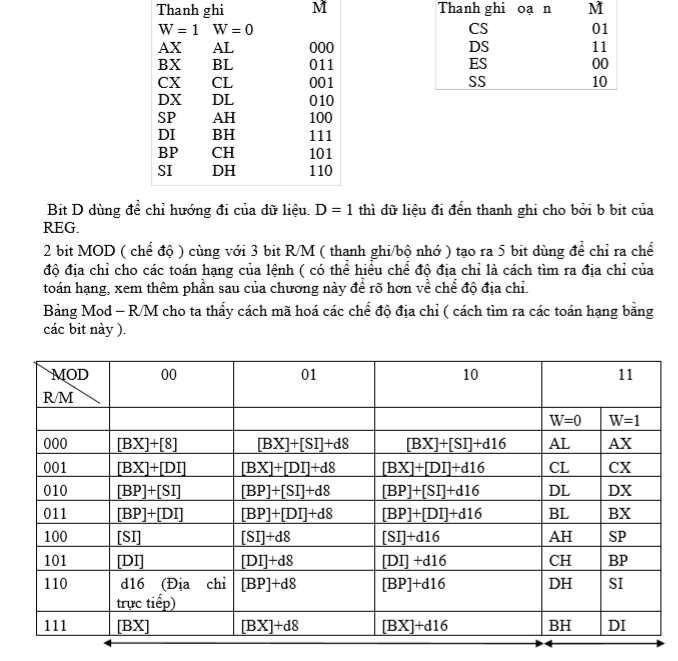
Với **OPCODE** : 100010 -> **MOV**

Với **D** :

**D = 0** : REG mã hóa cho nguồn -> Giải mã nguồn trước

**D = 1** : REG mã hóa cho đích -> Giải mã đích trước

Tiến hành tra bảng sau :



Vì **D = 1** nên ta **giải mã đích trước**

Với **W = 0 + REG = 110** -> REG là **DH ->** Điền DH vào đích

Với **MOD = 01 + R/M = 111 -> [BX] + d8**, mà **d8** ở đây, chính là phần kí tự hexa thừa mà vữa nãy ta không đổi ra nhị phần, nên cuối cùng sẽ là [BX] + 5Dh. Điền [BX] + 5Dh vào nguồn

Vậy : **8A775Dh** giải mã ra câu lệnh **‘MOV DH,[BX]+5Dh’**

**1.2.** **8BDAh**

100010 1 1 11 011 010

Tương tự như trên, sau khi giải mã, ta thu được câu lệnh **MOV BX,DX**

**1.3. 88B7A482h**

100010 0 0 10 110 111 A482

Ở câu lệnh này có đôi chút khác biệt, ta có tới 8 kí tự hexa, ta sẽ đổi 4 kí tự đầu tiên sang nhị phân, 4 kí tự còn lại không đổi, nhưng hãy đảo vị trí hai kí tự đầu và hai kí tự cuối cho nhau rồi điền vào phần d16

Cụ thể : **A4 82** sẽ viết lại thành **82 A4 (LƯU Ý KĨ ĐIỀU NÀY)**

Cũng tương tự như câu trên, sau khi giải mã, ta thu được câu lệnh :

**MOV [BX] + d16, DH**

Điền d16 là 82A4 -> Câu lệnh cuối cùng là : **MOV [BX] + 82A4h, DH**

**OK, TỔNG KẾT, BA CÂU LỆNH LẦN LƯỢT LÀ :**

**1. MOV DH, [BX]+5Dh**

**2. MOV BX,DX**

**3. MOV [BX] + 82A4h, DH**

**Bước 2 : Đổi địa chỉ logic sang địa chỉ vật lý, phân tích giá trị sau các câu lệnh**

Ở câu lệnh 1 và 3, có xuất hiện hai địa chỉ logic

+ [BX] + 5Dh (1)

+ [BX] + 82A4h (2)

Ta sẽ tiến hành đổi (1) sang địa chỉ vật lý

Trước hết, [BX] có nghĩa là địa chỉ ô nhớ được trỏ bởi thanh ghi BX và [BX] = DS:BX

Theo đề bài : BX = 83ADh, DS = 9F2Ch -> DS : BX = 9F2Ch : 83ADh

Để tính DS : BX, ta thêm 0 vào cuối DS rồi cộng với BX

Cụ thể : 9F2Ch:83ADh -> 9F2C0h + 83ADh = A766Dh = [BX]

Sau đó cộng nốt với 5Dh : A766Dh + 5Dh = A76CAh

**-> [BX] + 5Dh = [A76CAh]**

Lúc này, câu lệnh số 1 được viết lại là : **MOV DH, [A76CAh]**

Lại theo đề bài : [A76CAh] = E6h

**Suy ra : DH = E6h**

**Kéo theo đó, giá trị của thanh ghi DX cũng bị thay đổi theo, cụ thể, giá trị DX đang là 923Ah, sau câu lệnh số 1, giá trị mới của DX là E63Ah**

**(Bởi mỗi thanh ghi đều có 2 phần : phần cao và phần thấp. Với DX = DH + DL)**

Với câu lệnh số 2 : **MOV BX,DX**

Sau câu lệnh này, giá trị của BX bị thay đổi, cụ thể là lấy giá trị của thanh ghi DX lưu vào thanh ghi BX

Giá trị của BX sau câu lệnh số 2 là : **E63Ah**

Với câu lệnh số 3 : **MOV** **[BX] + 82A4h, DH**

Ta lại tiến hành đổi địa chỉ logic sang địa chỉ vật lý, với giá trị mới của BX là E63Ah

Tương tự như trên, [BX] + 82A4h = [B5B9Eh]

Câu lệnh số 3 được viết lại là : **MOV** **[B5B9Eh], DH**

Theo câu lệnh số 1**, DH = E6h -> [B5B9Eh] = E6h**

**Kết luận : Giá trị được lưu tại ô nhớ có địa chỉ vật lý là B5B9Eh là E6h**

**HẾT BÀI !**

**CÂU 6 :**

*8A4618h*

*100010 1 0 01 000 110 18h*

*MOV AL [BP]+18h*

*[BP] = SS:BP = 1F34h : 86BCh*

*1F340h + 86BC = 279FCh + 18h = 27A14h*

*MOV AL,[27A14h] = 35h -> AL = 35h*

*8AE8h*

*100010 1 0 11 101 000 b*

*MOV CH AL = 35h*

*-> CH = 35h -> CX = 35DEh*

*888EC21Ah*

*100010 0 0 10 001 110 C21Ah*

*MOV [BP]+1AC2h,CL*

*279FCh + 1AC2h = 294BE*

*-> MOV [294BEh],CL = DEh*

*-> [294BEh] = DEh*

**DONE !**

**MỘT VÀI LƯU Ý KHI LÀM DẠNG BÀI NÀY :**

*1. Làm thật cẩn thận, chậm mà chắc*

*2. Hãy theo dõi kĩ sự thay đổi về giá trị của các thanh ghi sau các câu lệnh, bởi giá trị thanh ghi của câu này lại liên quan tới câu sau, sai một ly là đi một dặm*

*3. Trong đề có nhắc tới địa chỉ các ô nhớ, trong quá trình tính toán, chắc chắn sẽ tính ra đúng các địa chỉ này, nếu tính toán mà ra địa chỉ lệch với đề bài, chắc chắn là anh em đã sai ở đâu đó, hãy check thật kĩ*