**ĐỀ THI CTMT&HN 07**

**Câu 1:**  
Giả sử 2 lệnh PUSH BX, PUSH CX được thực hiện ngay trước khi debug hiển thị:  
AX=3520 BX=00FF CX=1234 DX=FFE2 SP=FF00  
BP=0000 SI=0000 DI=0000 DS=0000 ES=0FF0  
SS=12EB CS=1F76 IP=010A NV UP EI PL NZ NA PO NC  
1F76 : 010A 58 POP AX  
a.[FONT=&quot] [/FONT]Cho biết nội dung CS, IP, SS, SP sau lệnh kế  
***CS=1F76, IP=010B, SS=12EB,SP=FF02***  
b.[FONT=&quot] [/FONT]Vẽ sơ đồ trình bày sự thay đổi đỉnh stack, chỉ rõ đỉnh stack cũ, mới, địa chỉ và nội dung các ô nhớ bị thay đổi.  
  
  
***FNguyen:*** *CSmoi=CScũ=1F76  
IPmới=IPcũ+1=010B  
SSmoi=SScu=12EB  
SPmoi=SPcu+2=FF02*  
  
  
  
**Câu 2 :**  
AX=3520 BX=00FF CX=1234 DX=FFE2 SP=FF00  
BP=0000 SI=0000 DI=0000 DS=0000 ES=0FF0  
SS=12EB CS=1F76 IP=010A NV UP EI PL NZ NA PO NC  
1F76 : 010A 082300 CALL 0130  
c.[FONT=&quot] [/FONT]Cho biết nội dung CS, IP, SS, SP sau lệnh kế  
***CS=1F76, IP=0130, SS=12EB, SP=FEFE***  
d.[FONT=&quot] [/FONT]Vẽ sơ đồ trình bày sự thay đổi đỉnh stack, chỉ rõ đỉnh stack cũ, mới, địa chỉ và nội dung các ô nhớ bị thay đổi.  
  
**Câu 3:**  
a.[FONT=&quot] [/FONT]Cho biết kết quả của đoạn chương trình sau (AL=?)  
Char table DB ‘0123456789ABCDEF’  
MOV AL,10  
MOV BX, OFFSET char table  
XLAT  
b.[FONT=&quot] [/FONT]Cho biết kết quả của thanh ghi AX dạng số hex sau khi thực hiện đoạn chương trình sau:  
MOV AX,9  
MOV BX,3  
ADD AL,BL  
AAA  
c.[FONT=&quot] [/FONT]AX=?  
Mem DW 8  
MOV AX,Mem  
SHL AX,1  
MOV BX, AX  
SHL AX,1  
SHL AX,1  
ADD AX,BX  
  
**Câu 4:**  
a.[FONT=&quot] [/FONT]AL=?  
MOV AX,5936H  
ADD AL,AH  
DAA  
b.[FONT=&quot] [/FONT]Var2=?  
Var1 DW 1234  
Var2 DW ?  
MOV AX,Var1  
SUB AX,0  
JS Signed  
JMP Next  
Signed:  
NEG AX  
Next:  
MOV Var2, AX  
c.[FONT=&quot] [/FONT]Cho biết kết quả  
Var DD 2345678  
MOV AX, word PTR Var  
MOV DX, word PTR Var+2  
MOV BX,10  
DIV BX  
  
**Câu 5:**  
a.[FONT=&quot] [/FONT]Thay thế các lệnh sau bằng một lệnh tương đương  
MOV BX, [SI]  
MOV AX, [SI+2]  
MOV DS,AX  
***LDS BX,SI***  
b.[FONT=&quot] [/FONT]DX=?, AX=?  
Mem16 DW -2000  
Mem32 DD 500000  
MOV AX,word PTR Mem32[0]  
MOV DX,word PTR Mem32[2]  
IDIV Mem16  
c.[FONT=&quot] [/FONT]AX=? sau khi thực hiện lệnh AAD và AAM  
MOV AX,0905H  
MOV BL,3  
AAD  
DIV BL  
AAM  
  
**Câu 6:**  
a.[FONT=&quot] [/FONT]Cho biết sau đoạn chương trình con, điều khiển sẽ trở về chương trình chính tại địa chỉ nào?  
…  
MOV AX, 0F00H  
PUSH AX  
MOV AX,0  
PUSH AX  
RETF  
b.[FONT=&quot] [/FONT]AX=?  
NAME DW ‘ALA’  
CLD  
LEA SI,NAME  
INC SI  
LODSW  
c.[FONT=&quot] [/FONT]Dem=?  
Mem DB ‘ACDE’  
DB 50 DUP(0)  
DB ‘ES0123SFS’  
DEM DW 0  
MOV CX,63  
MOV BX,OFFSET Mem  
MOV DEM,0  
CONT:  
CMP BYTE PTR[BX],’S’  
JNZ Notmatch  
INC DEM  
Notmatch:  
INC BX  
LOOP CONT  
  
**Câu 7:**  
a.[FONT=&quot] [/FONT]ARR=?(giả sử DS và ES đã chứa đoạn dữ liệu)  
ARR DW 10,20,40,50,60  
STD  
LEA SI,ARR+8H  
LEA DI,ARR+AH  
MOV CX,3  
REP MOVSW  
MOV WORD PTR [DI],30  
***ARR = 10,20,30,40,50,60***  
b.[FONT=&quot] [/FONT]CX=?  
MOV DX,1234H  
MOV BX,3039H  
MOV AL,BL  
ADD AL,DL  
DAA  
MOV CL,AL  
MOV AL,BH  
ADC AL,DH  
DAA  
MOV CH,AL  
  
**Câu 8:**  
a.[FONT=&quot] [/FONT]AX=?  
VAL DB 5  
MOV AX,0  
MOV AL,-48  
IDIV VAL  
b.[FONT=&quot] [/FONT]BX=?  
MOV BX,3235H  
AND BX,0F0FH  
MOV DX,BX  
SHL BH,4  
OR BL,BH  
XOR BH,BH  
  
**Câu 9:**  
Mô tả ảnh hưởng của các lệnh sau đây đối với dữ liệu trong data segment dưới đây:

ADDRESS

CONTENTS

1012

0005

1010

1067

100E

0025

100C

4156

A.[FONT=&quot] [/FONT]MOV [1012H],DX DX=FE00H  
***[1012]=00, [1013]=FE***  
B.[FONT=&quot] [/FONT]MOV AX,100EH  
***AX=100EH***  
C.[FONT=&quot] [/FONT]ADD AX,[BX+DI] AX=5, BX=100E, DI=2  
***AX=106CH***  
D.[FONT=&quot] [/FONT]MOV AX,LIST[BX+SI] AX=1000H , LIST=100CH, BX=0,SI=4  
***AX=1067H***  
E.[FONT=&quot] [/FONT]ADD TABLE[BX+DI-5] CX=3, TABLE=100EH, BX=-0004H, DI=000AH  
***[100F]=00, [1010]=03***  
  
**Câu 10:** Cho một vi xử lý 8080(20 bit địa chỉ, 8 bit dữ liệu) . Dùng kĩ thuật giải mã địa chỉ để ánh xạ 32 KB bit cao nhất vào 8 chip EFROM 4K x 8. Vẽ mạch và tính toán địa chỉ đầu cuối cho mỗi EFROM.

**Câu 1:**  
a) ***CS=1F76, IP=010B, SS=12EB,SP=FF02***  
***Chú ý: Khi thực hiện POP thì***  
***CS mới = CS cũ.***  
***IP mới = IP cũ + số byte của lệnh .***  
***SS mới = SS cũ.***  
***SP mới = SP cũ + số byte dữ liệu lấy ra.***  
  
b) Sơ đồ :  
  
**Câu 2:**  
  
a) ***CS=1F76, IP=0130, SS=12EB, SP=FEFE***  
***Chú ý: Do thực thi lệnh CALL gọi chương trình con với dạng target là kiểu near nên: đầu tiên nó PUSH IP sau đó Nạp địa chỉ offset của target vào IP, chuyển điều khiển tới chương trình con.***  
+ CS mới = CS cũ.  
+ IP mới = địa chỉ offset của target .   
+ SS mới = SS cũ.  
+ Sp mới = SP cũ - só byte của dữ liệu của IP. Do Push IP.  
b) Sơ đồ stack:  
  
**Câu 3:**  
**a) Lệnh XLAT nội dung của ô nhớ AL ứng với dư liệu trong bảng 8 bit DS:BX với vị trí AL.**  
**Vì vậy ở đây ta có DS:BX chứa bảng tra là ‘**0123456789ABCDEF’ **, và vị trí tra lấy dữ liệu chứa trong AL là 10 nên sau khi thực hiện XLAT sẽ có**   
**Temp = AL + BX**  
**AL= DS : [temp]**  
**Mà AL = 10H , và vd BX= 200H thì temp = 210H, mà chuỗi chỉ dài tớ 20F => AL chứa nội dung của ô có địa chỉ DS:0210 .**  
**Các bạn lưu ý : thứ tự của ký tự trong chuỗi sẽ từ 0200-0201-0202-0203-0204-0205-0206-0207-0208-0209-020A-020B-020C-020D-020E-020F.Vì thế kết quả không là 1 ký tự trong chuỗi.**  
**Cái này làm theo debug chứ ko chạy chương trình .**  
**b) Kết quả thanh ghi AX sẽ là AX=0102 . Vì khi thực hiện ADD ta sẽ được AX = 0012 theo dạng BCD hay AX=000C.Sau khi thực hiện AAA để chỉnh lại ta được AX= 0102**  
**CT chuyển đổi của AAA -> xem lệnh AAA.**  
**c) AX = 0050**  
**Đầu tiên AX = 0080 lưu dạng BCD, lệnh SHL(1) thì AX=0010 dịch trái 1, sau đó sao chép vào BX, tiếp tục dịch trái 2 lần AX ta có 0020 rồi 0040 . Cuối cùng cộng AX và BX lại 0010+0040 -> 0050**  
**AX = 0000 0000 0000 1000**  
**AX = 0000 0000 0001 0000 -> BX**  
**AX = 0000 0000 0010 0000**  
**AX = 0000 0000 0100 0000**  
**BX = 0000 0000 0001 0000**  
**AX = 0000 0000 0101 0000 = 0050**  
  
  
**Câu 4:**  
**a) AL = 22H và CF = 1**  
**AX = 5963 = 0101 1001 0110 0011**  
**AH = 0101 1001**  
**AL = 0110 0011**  
**AL = 10010 0010 -> AL = 0010 0010 và CF = 1.**  
**b) Var2 = 1234. Chuyển dl 1234 -> Ax sau đó thực hiện AX <- Ax -0 , sau đó thực hiện so sánh xem có âm ko , nếu có làm cho dương lại còn ko thì di chuyển đến Next thực hiện di chuyển AX-> Var2.**  
**c) Lỗi tràn “Divide Overflow”. Do thương số vượt quá giới hạn AX nên CPU thông báo lỗi tràn trong phép chia.(2345678 / 10 -> 234567 vượt quá AX ).**  
  
  
**Câu 5:**  
  
**Giải:**  
**a) LDS BX,SI . Nạp phần cao của SI vào DS (Mov ax,[si+2] ; mov ds,ax) , nạp phần thấp vào BX (mov bx,[si]).**  
**b)**   
**c) AX = 0301.**   
**Đầu tiên AX = 0905 = 0000 1001 0000 0101**  
**AH = 09 = 0000 1001**  
**AL = 05 = 0000 0101**  
**BL = 3 = 0000 0011**  
**Thực hiện lệnh AAD để hiệu chỉnh trước khi chia 2 số BCD.**  
**AL <- AH \* 10 +AL ; AH<- 0**  
**=> AH = 0000 0000**  
**AL = 0101 1111**  
**DIV BL thực hiện chia AL <- AL / BL ta có**  
**=> AH = 0000 0000**  
**AL = 0001 1111.**  
**Thực hiện lệnh AAM để hiệu chỉnh kết quả.**  
**AH <- AL div 10 ; AL <- AL mod 10;**  
**AH = 0000 0011 = 03**  
**AL = 0000 0001 = 01**  
  
**Câu 6:**  
**Giải :**  
**a) Chương trình con sẽ quay về chương trìn chính tại vị trí là 0F00:0000**  
**Vì lệnh RETF là lệnh quay về từ lệnh gọi xa và POP nên nó sẽ thực hiện như sau :**   
**POP AX lấy dữ liệu stack ra đưa vào AX (đây là offset 0000 ) ,**  
**Tiếp POP CS lấy dữ liệu stack ra đưa vào CS , khôi phục lại segment chương trình chính trước khi gọi ct con, CS = 0F00.**  
**sau đó thực hiện lệnh JMP AX tức nhảy ko điều kiện tới AX hay Offset 0000.**  
**Cuối cùng trả lại chương trình chính tại địa chỉ là 0F00:0000 .**   
  
**b)**