CS/ECE/EEE/INSTR F241 – MICROPROCESSOR PROGRAMMING & INTERFACING

MODULE 4: 80X86 INSTRUCTION SET

QUESTIONS

ANUPAMA KR
BITS, PILANI – KK BIRLA GOA CAMPUS

Q1. IWrite an ALP that will examine a set of 20 memory locations that have alphabets and count the number of vowels. The alphabets are store from memory location ***alph1*** and the count of the vowels must be stored in location ***vcnt***.

Q2. Write an ALP that will copy a set of 10 bytes from memory location ***loc1*** to memory location ***loc2*** in the reverse order.

Q3. An interleaved string is stored from displacement '***istr1***'. The size of the interleaved string is stored in location '***cnt1***'. Write an ALP that will separate the interleaved string into two strings as shown. If the interleaved string is "hmeilclroo" it should be separated as two strings "hello" and "micro". You can assume that the strings to be separated will be of equal size. [note – there is no need to use string instructions for this]

Q5. An array of data is stored in data segment starting from ***ARRAY***. The number of elements in the array is stored in location ***COUNT***. Write an ALP to find the minimum number and the displacement at which it is stored. The number must be stored in memory location ***MIN*** and the address of the number in memory location ***MINADDR***.

Q6. An array of data is stored in data segment starting from ***ARRAY***. The number of elements in the array is stored in location COUNT. Write an ALP to find the maximum number and the displacement at which it is stored. The number must be stored in memory location ***MAX*** and the address of the number in memory location ***MAXADDR.***

Q7. An array of data is stored in data segment starting at ***ARRAY***. The number of elements in the array is stored in location ***COUNT***. Write a program to count the number of occurrences of the element in ***CODE1*** in the ***ARRAY*** and store this result in location ***RESULT***.

Q8. An array of data is stored in data segment starting from ***ARRAY***. The number of elements in the array is stored in location ***COUNT***. Write an ALP to arrange these number in the ascending order. Also store the value of the second largest number in the array in location ***SECMAX***.

Q9. Write an 80486 ALP that will add the two nibbles in a data byte together and if there is a carry in the nibble addition it will write 'C' into a memory location. If there is no carry in nibble addition it will write 'N' into the location. This has to be done on array of 25d data bytes stored from location dat1. The result for each byte must be written from location car1.

For E.g. If

dat1 db 45h,89h,27h, 0F2h, 3Eh and so on

The program has to add 4 and 5 in case of first byte and there will be no carry in adding the two nibbles (4-bit addition) so N has to be written into car1 and if case of 2nd data 8 and 9 will be added and result will have a carry in the nibble addition so C will be written into next location.

Hence after the program is executed the location

car1 will have the values 'N', 'C', 'N','C','C' and so on

Q10. Write a program that examines the contents of 50 memory locations that has ASCII characters and counts the number of Numerals, Capital alphabets, Small Alphabets and stores them in memory locations labelled ***NUMS, CAPS1, SMA1***.

Q11. Write a program that will count the number of 0's in a 32-bit data stored in location ***dat1*** and store the resultant count in location ***res1***.

Q12. Write an 80486 ALP with a subroutine '***sub1***' that will count number of odd positive, odd negative, even positive and even negative 32-bit numbers. The data that is to be analysed is stored in memory location 'in1' and results will be stored in locations '***oddpos***', '***oddneg***', '***evenpos***','***evenneg***' respectively. The total count of data available is in location '***cnt1***' and cannot exceed 100. The subroutine should do the categorization and counting while the main program only passes the number as a parameter using BX as the pointer.

Q13. Write an ALP that will search for character in a set of 100 locations. If character 'A' occurs then it must be replaced by character 'Z' – use cmpxchg instruction

Q14. Write a program that will examine a word array ***arrayw*** in memory if the word is even then the program must sign extend the data and store it in a double word array – ***arrayd*** - if the word is odd the word data must be zero extended into double word data.

Q15. Write a program that will convert a 32-bit data stored in little endian format to bigendian format.

Redo this for any array – using the previous code you have written as

- Sub-program
- Macro

Q16. Write a ALP that will find out whether data stored in loc1 is a palindrome. The size of the palindrome is stored in location cloc1.

Make this as a sub program –that can be accesses by a main program that handles array of numbers. The count of the palindrome must be stored in location ***dloc1.*** Repeat the same using macros.

Q17. Write a program that will set the trap Flag – do not use PUSH & POP Instructions.

Q18. Write an ALP to scan a string stored from memory location labelled *ARR1* for blank spaces and replace every blank space in the string with a ^ The size of the array is stored at location with label *CNT*.

Q19. Write an 80486 ALP that will examine a series of memory locations for small alphabets. If a memory location has a small alphabet it will convert it into capitals. If the memory location does not have a small alphabet it will not modify the contents of the memory location. The series of memory location to be examined start at ***alph1***. The count of memory locations to be examined is stored in ***cnt1*** and will not exceed 1000_d. The checking and the conversion of one small alphabet to one capital alphabet must be done using a macro called CAPSON.

Q20. Two arrays of unsigned 8 -bit data numbers are stored from location *arr1* and *arr2*. Write a program that will add the contents of *arr1* with *arr2* and store the addition result including the carry in an unsigned 16-bit array *arr3*. The count of data in *arr1* and *arr2* is 5.

For e.g. if the data in arr1 is	45h,	82h,	91h,	73h,	13h
And the data in arr2 is	20h,	7fh,	33h,	8eh,	45h
The result in arr3 will be	0065h,	0101h,	00c4h,	0101h,	0058h

Q21. A set of signed 8 -bit data is stored from location *dat1*. The count of the data is available in location *cnt1*. Write an ALP that will check whether a number is negative, if the number is negative finds the 2's complement of the number and stores it back in the same location. If number is positive there will be no change. You can assume that count of data will not exceed 100.

For e.g. if the data is 45h, 82h, 91h, 23h, 13h

The ALP must convert the data to 45h, 7eh, 6fh, 23h, 13h

Q22. Write an ALP to swap the contents of two 10-byte arrays in memory. The first array is stored at location with label *ARR1*. The second array is stored at location with label *ARR2*

Q23. Given below is an 80x86 assembly program segment

```
.Model Tiny
 .486
. DATA
L1          DB          'a', '$', '*', 'h', 56H, 12
L2          DD          0AABBCCDEH
L4          EQU         0A0DH
L5          DB          'WHERE',  3  DUP ('$')
            DW          3 DUP (0A0DH)
L6          DW          100 DUP('0')
S1          DW          ?
.CODE
.STARTUP
            LEA         SP, S1
            LEA         SI, L5
            ADD         SI,8
            MOV         BX,OFFSET L1
            XOR         CX, CX
            MOV         AL, [BX + 4]
            MOV         CX, L4
            PUSH        BX
            ADD         CL, AL
            POP         CX
            CMP         [SI],CX
.EXIT
END
```

(a) Write the contents of memory in data segment that result from data declarations in the program given in the tabular format given below. **(i.e., for 40 locations in data segment assume starting from offset 0120_H to 0147_H)** (You may use 'A' to represent ASCII byte for the character A. If the contents cannot be determined put an 'X' in the box. **<u>All values except for ASCII values must be in hexadecimal</u>**)

DS:0120_H								
DS:0128_H								
DS:0130_H								
DS:0138_H								
DS:0140_H								

(b) For the code section of the program. Fill in the table below. (You can assume that all status flags are cleared initially). You only need to show contents of only registers that are affected (If no register affected just enter none.) **<u>Values of registers must be given in hexadecimal (unless ASCII).</u>**

Instruction	Register contents	OF	SF	CF	ZF
LEA SP, S1					
LEA SI, L5					
ADD SI, 8					
MOV BX,OFFSET L1					
XOR CX, CX					
MOV AL, [BX + 4]					
MOV CX, L4					
PUSH BX					
ADD CL,AL					
POP CX					
CMP [SI],CX					

Q24. Replace the following program segments by **a single instruction** of 80486. You can assume that all flags (except Trap and Interrupt) are reset at the beginning of each of these program segment

[Clarification: Each program segment achieves a certain final result. You need to give a single instruction that will achieve the same result. The single instruction needs only achieve the final result]

	Program	Instruction		Program	Instruction
A	**PUSH AX** **PUSH BX** **POP AX** **POP BX**		**B**	**PUSHF** **PUSH BP** **MOV BP,SP** **MOV AH,[BP+2]** **POP BP** **POPF**	
C	**CMP EBX,EAX** **JNE X1** **MOV EBX,ECX** **JMP X2** **X1: MOV EAX,EBX** **X2:**		**D**	**MOV [0200$_H$],ESP*** **PUSH EAX** **PUSH ECX** **PUSH EDX** **PUSH EBX** **PUSH DWORD PTR[0200$_H$]** **PUSH EBP** **PUSH ESI** **PUSH EDI**	
E	**PUSHF** **MOV BH,FF$_H$** **CMP BL,0** **JL X1** **NOT BH** **X1: POPF**		**F**	**PUSHF** **PUSH AX** **MOV AX,[SI]** **MOV ES: [DI],AX** **POP AX** **INC SI** **INC SI** **INC DI** **INC DI** **POPF**	

***DS: [0200$_H$] – is just a temp location – what happens to it does not matter in the final result.**

Q25. What will be the effect of executing the following code snippet on an 8086 processor?

MOV BX, 0FFFF$_H$
AND BX, 0700$_H$
PUSH BX
POPF

Q26. Replace the following program segments by **a single instruction** of 80486

	Program	Instruction		Program	Instruction
A	**PUSH SI** **ADD SI,DI** **POP DI**		B	**JNC X1** **ADD BX,1** **X1: ADD BX,CX**	
C	**MOV EBX,EAX** **MOV ECX,EAX** **MOV EDX,EAX** **AND EAX,000000FFH** **AND EBX,0000FF00H** **AND ECX,00FF0000H** **AND EDX,FF000000H** **ROL EDX, 8** **ROR ECX,8** **ROL EBX,8** **ROR EAX,8** **OR EAX,EBX** **OR EAX,ECX** **OR EAX,EDX**		D	**BT AX,15** **JC X1** **AND EAX,0000FFFFH** **JMP X2** **X1: OR EAX, FFFF0000H** **X2:**	

Q27. Given below is an 8086 assembly program.

```
.Model          Tiny
. DATA
  DAT1          DB      45H, 54H, 46H
  P1            EQU     97H
  DAT2          DW      23F8H, 2435H
  DAT3          DB      'INTER'
  DAT4          DB      6 DUP (122)
  DAT5          DB      3 DUP (?)
  DAT6          DW      33H
  DAT7          DB      0FH
.CODE
.STARTUP
                MOV     AL, DAT1+1
                ADD     AL,DAT4
                CBW
                MOV     BX, 12AH
                MOV     CX, [BX+4]
                XOR     CH,P1
```

(a) Write the contents of memory in data segment that result from data declarations in the above program in the tabular format given below. **(i.e., for 24 locations in data segment assume starting from offset 0118$_H$ to 012F$_H$)** (You may use 'A' to represent ASCII byte for the character A. If the contents cannot be determined put a 'X' in the box. **All values except for ASCII values must be in hexadecimal**)

DS:0118$_H$								
DS:0120$_H$								
DS:0128$_H$								

(b) For the code section of the above program. Fill in the table below. (You can assume that all status flags are cleared initially). You only need to show contents of registers that are affected. **Values must of registers must be given in hexadecimal.**

Instruction	Register contents	Addressing Mode	ACF	OF	SF	PF	CF	ZF
MOV AL,DAT1+1								
ADD AL,DAT4								
CBW								
MOV BX, 12A$_H$								
MOV CX,[BX+4]								
XOR CH,P1								

Q28. If AX=FFFFH and CL =02H . What will happen if you execute the instruction DIV CL

Q29. For an 80386 processor write a single instruction that will swap the nibbles of the AL register

Q30. What is the only status flag of 80386 whose content you cannot check using an unconditional jump instruction?

Q31. Suppose that SP=3000H , SI=0250H , BX=2345H , AX=6789H . Assume that initial value of Flag register is 0231H . Following instructions are executed in a 80386 processor. Mention the values present in all the register after execution of the following instructions

```
PUSH AX
PUSH BX
PUSHF
PUSH 0987H
PUSH DI
PUSH SI
POP AX
POP BX
POPF
POP DI
POP [SI]
```

Q32. The contents of SI=0003h, AX=0001h After execution of XADD SI, AX on an 80386 processor what will be the contents of SI and AX registers?

Q33. Write a program snippet to implement jump if Auxiliary carry to displacement 32h with respect to IP.

Q34. What will happen if you end an ISR using RET instead of IRET ?

Q35. If you want to convert signed word in AX register of 80386 into a signed double word to be stored in EAX which instruction will you use?

Q36. What is the difference between putting .386 directive before. model tiny declaration and putting .386 directive after. model tiny declaration?

Q37. Why is Interrupt flag disabled on an entry into an ISR?

Q38. Write an Assembly language program segment to do the following function: JZ $2FF_H$.

Q39. The following hypothetical program runs in 8086 .What will be the contents of registers AX, BX and SP after execution. Assume initially AX=0000, BX=0000, SP=$FFFE_H$.

MOV AX, 2037_H

MOV BX, 0542_H

MOV SS, AX

MOV SP, BX

PUSH AX

PUSH BX

POP AX

ADD AX,BX

Q40. What will be the result of Executing the following code snippet?

```
LAHF
AND AH,10H
JZ 50H
```