ASSEMBLY LANGUAGE PROGRAMMING (CS401)

SUBJECTIVE SOLVED FOR MID TERM

❖ **Explain the process of ADC?**

Normal addition has two operands and the second operand is added to the first operand. However ADC has three operands. The third implied operand is the carry flag. The ADC instruction is specifically placed for extending the capability of ADD. Further more consider an instruction "ADC AX, BX." Normal addition would have just added BX to AX, however ADC first adds the carry flag to AX and then adds BX to AX. Therefore the last carry is also included in the result. The lower halves of the two numbers to be added are first added with a normal addition. For the upper halves a normal addition would lose track of a possible carry from the lower halves and the answer would be wrong. If a carry was generated it should go to the upper half. Therefore the upper halves are added with an addition with carry instruction.

❖ **What is the difference between LES and LDS instructions ?**

The string instructions need source and destination in the form of a segment offset pair. LES and LDS load a segment register and a general purpose register from two consecutive memory locations. LES loads ES while LDS loads DS. Both instructions has two parameters, one is the general purpose register to be loaded and the other is the memory location from which to load these registers. The major application of these instructions is when a subroutine receives a segment offset pair as an argument and the pair is to be loaded in a segment and an offset register.

❖ **Describe local variables.**

Another important role of the stack is in the creation of local variables that are only needed while the subroutine is in execution and not afterwards. They should not take permanent space like global variables. Local variables should be created when the subroutine is called and discarded afterwards. So that the spaced used by them can be reused for the local variables of another subroutine. They only have meaning inside the subroutine and no meaning outside it.

The most convenient place to store these variables is the stack. We need some special manipulation of the stack for this task. We need to produce a gap in the stack for our variables. This is explained with the help of the swapflag in the bubble sort example.

The swapflag we have declared as a word occupying space permanently is only needed by the bubble sort subroutine and should be a local variable. Actually the variable was introduced with the intent of making it a local variable at this time. The stack pointer will be decremented by an extra two bytes thereby producing a gap in which a word can reside. This gap will be used for our temporary, local, or automatic variable; however we name it. We can decrement it as much as we want producing the desired space, however the decrement must be by an even number, as the unit of stack operation is a word. In our case we needed just one word. Also the most convenient position for this gap is immediately after saving the value of SP in BP. So that the same base pointer can be used to access the local variables as well; this time using negative offsets. The standard way to start a subroutine which needs to access parameters and has local variables is as under.

```
push bp
mov bp, sp
sub sp, 2
```

The gap could have been created with a dummy push, but the subtraction makes it clear that the value pushed is not important and the gap will be used for our local variable. Also gap of any size can be created in a single instruction with subtraction. The parameters can still be accessed at bp+4 and bp+6 and the

swapflag can be accessed at bp-2. The subtraction in SP was after taking the snapshot; therefore BP is above the parameters but below the local variables. The parameters are therefore accessed using positive offsets from BP and the local variables are accessed using negative offsets.

- **Explain LES and LDS**

The string instructions need source and destination in the form of a segment offset pair. LES and LDS load a segment register and a general purpose register from two consecutive memory locations. LES loads ES while LDS loads DS. Both instructions has two parameters, one is the general purpose register to be loaded and the other is the memory location from which to load these registers. The major application of these instructions is when a subroutine receives a segment offset pair as an argument and the pair is to be loaded in a segment and an offset register.

- **Explain the use of TEST instruction.**

The test instruction is used for bit testing. BX holds the mask and in every next iteration it is shifting left, as our concerned bit is now the next bit.

- **The Formula to scroll up the screen**

rep movsw **scroll up**

```
scrollup: push bp
mov bp,sp
push ax
push cx
push si
push di
push es
push ds
mov ax, 80 ; load chars per row in ax
mul byte [bp+4]          ; calculate source position
mov si, ax               ; load source position in si
push si                  ; save position for later use
shl si, 1                ; convert to byte offset
mov cx, 2000             ; number of screen locations
sub cx, ax               ; count of words to move
mov ax, 0xb800
mov es, ax               ; point es to video base
mov ds, ax               ; point ds to video base
xor di, di               ; point di to top left column
cld                      ; set auto increment mode
rep movsw                ; scroll up
mov ax, 0x0720           ; space in normal attribute
pop cx                   ; count of positions to clear
rep stosw                ; clear the scrolled space
pop ds
pop es
pop di
pop si
pop cx
pop ax
pop bp
ret 2
```

- **What is difference between SHR and SAR instructions?**

SHR

The SHR inserts a zero from the left and moves every bit one position to the right and copy the rightmost bit in the carry flag.

SAR

The SAR shift every bit one place to the right with a copy of the most significant bit left at the most significant place. The bit dropped from the right is caught in the carry basket. The sign bit is retained in this operation.

- **Write a subroutine to calculate the string length.?**

```
subroutine to calculate the length of a string
; takes the segment and offset of a string as parameters
strlen: push bp
mov bp,sp
push es
push cx
push di
les di, [bp+4]          ; point es:di to string
mov cx, 0xffff          ; load maximum number in cx
xor al, al              ; load a zero in al
repne scasb             ; find zero in the string
mov ax, 0xffff          ; load maximum number in ax
sub ax, cx              ; find change in cx
dec ax                  ; exclude null from length
pop di
pop cx
pop es
pop bp
ret 4
```

- **Explain how extended shifting is performed**

Using our basic shifting and rotation instructions we can effectively shift a 32bit number in memory word by word. We cannot shift the whole number at once since our architecture is limited to word operations. The algorithm we use consists of just two instructions and we name it extended shifting.

```
num1: dd 40000
shl word [num1], 1
rcl word [num1+2], 1
```

The DD directive reserves a 32bit space in memory; however the value we placed there will fit in 16bits. So we can safely shift the number left 16 times.

The least significant word is accessible at num1 and the most significant word is accessible at num1+2.

The two instructions are carefully crafted such that the first one shifts the lower word towards the left and the most significant bit of that word is dropped in carry. With the next instruction we push that dropped bit into the least significant bit of the next word effectively joining the two 16bit words.

The final carry after the second instruction will be the most significant bit of the higher word, which for this number will always be zero.

- **What is the Difference between CALL and RET**

The CALL instruction allows temporary diversion and therefore reusability of code. The word return holds in its meaning that we are to return from where we came and need no explicit destination.

Therefore RET takes no arguments and transfers control back to the instruction following the CALL that took us in this subroutine.

- **IF DF=0 what its represent and IF DF=1 what its represent ?**

The direction of movement is controlled with the Direction Flag (DF) in the flags register. If this flag is cleared DF=0, the direction is from lower addresses towards higher addresses and if this flag is set DF=1, the direction is from higher addresses to lower addresses. If DF is cleared, DF = 0 this is called the autoincrement mode of string instruction, and if DF is set, DF=1, this is called the autodecrement mode. There are two instructions to set and clear the direction flag.

- **What's your understanding about Incrementing and Decrementing Stack?**

Whenever an element is pushed on the stack SP is decremented by two and whenever an element is popped on the stack SP is incremented by two.

A decrementing stack moves from higher addresses to lower addresses as elements are added in it while an incrementing stack moves from lower addresses to higher addresses as elements are added.
As the 8088 stack works on word sized elements. Single bytes cannot be pushed or popped from the stack.

- **Is it necessary to provide the segment and offset address in case of FAR jump ?**

Segment and offset must be given to a far jump. Because, sometimes we may need to go from one code segment to another, and near and short jumps cannot take us there. Far jump must be used and a two byte segment and a two byte offset are given to it. It loads CS with the segment part and IP with the offset part.

- **Describe the push function**

The push function copies the operand on to the stack. When an item is pushed on a decrement stack, top of the stack is first decrement and element is then copied into this space

- **How string instruction work on block of data**

In a string instructions, block have a start and end. Instructions can work from the start toward to end and from the end towards start. They can work in both directions and the have to be allowed to work in both directions; otherwise some operations with overlapping blocks become not possible.

- **Why we need to clear the stack?**

When the parameters by pushed for subroutine are waste after the subroutine return. They must be clear from the stack be caller and the callee.

- **Difference b/w REP and REPE and (REPNE)**

The REP allow instructions to operate on a number of data elements in a one instructions, REPE repeats the instructions string instructions while the zero flash is set and REPNE repeat while not equal or repeat the instruction while the zero flag is not set

- **Explain all characteristics of SCAS instruction**

1. SCAS instructions has variety of functions it compare a source of byte or word in register AL or AX with destination string address by ES:DI and updating flags.
2. It is used to locate equality or in-equality in a string
3. SCAS is bit different from the other instructions.
4. REPE and REPNE are used with this type of instructions
5. This is more like CMP instructions that it does subtraction of the operands.

- **DESCIRBE LOCAL VARIABLE**

Local variables are created by STACK and these variables are needed when subroutine are in execution and not afterwards. They do not take place like global variables. They are temporary not permanent. Local variables are created when the subroutine is call. There meaning is in subroutine not outside. There most

convenient place is stack. Special manipulation is needed for this task. Same base pointer can be used to access the local variables.

- **Why REP prefix is not generally use in LODS instructions**

REP prefix use with LODS is not meaningful as only last value loaded will be remaining in register. It is use in loop paired with STOS

- **Define the NOT operator**

Not operator inverts the bits of byte or word operand. This is single operand instructions. And invert the result into 1's complement form.

- **Differentiate b/w Intra Segment and Inter Segment:**

Out of line procedures in the temporary division, the concept of the round about near calls are called Intra Segment. On the other hand far calls are called inter-segment class.

- **Explain the use of TEST instruction.**

The test instruction is used for bit testing. BX holds the mask and in every next iteration it is shifting left, as our concerned bit is now the next bit.

- **Explain LES and LDS:**

LES and LDS load a segment register and a general purpose register from two consecutive memory locations. LES loads ES while LDS loads DS. Instructions have two parameters, one is the general purpose register to be loaded and the other is the memory location from which to load these registers. There major application of these instructions is when a subroutine receives a segment offset pair as an argument and the pair is to be loaded in a segment and an offset register.

- **Describe MOVS and CMPS instructions**

The MOVS instruction is a byte transfer or word from the source location D:SI to the destination. ES: DI and updated SI and DI to point to the next locations. It is use to move block of memory. Dif plays important role in case of overlapping blocks And CMPS instructions subtracts the source location DS: SI from the destination location ES: DI. Source and the destination are unaffected.

- **Explain MUL instruction in both cases (i) if the source operand is byte (ii) if the source operand is a word**

MUL performs unsigned multiplication of the source operand and the accumulator. In the case of source operand is a byte, then it is multiply by the register AL and the double length result will be return into AH and AL. On the other side if the source operand is a word, then it is multiplied by the register AX, and the double-length result is returned in registers DX and X.

- **IF DF=0 what its represent and IF DF=1 what its represent?**

We can control direction with Direction Flag (DF) in the flag Register. If the DF is value with it will be auto increment. and if the flag Register DF=1 it can set auto decrement.

- **Purpose of INT 1**

Int 1 is Single step Interrupt, This interrupt is used in debugging with trap Flag. If the trap flag is set the single step interrupt is generated after every instructions.

- **Write any two control instructions?……**

These are instructions that control the program execution and flow by playing with the instruction pointer and altering its normal behavior to point to the next instruction. Some examples are:

```
cmp  ax, 0
jne  1234
```

We are changing the program flow to the instruction at 1234 address if the condition that we checked becomes true.

```
dec  cl              ; decrement bit count
 jnz  checkbit         ; repeat if bits left
```

q=18…RET instruction…

The word return holds in its meaning
that we are to return from where we came and need no explicit destination.
Therefore RET takes no arguments and transfers control back to the
instruction following the CALL that took us in this subroutine.

- **ROR instructions….**

In the rotate right operation every bit moves one position to the right and the bit dropped from the right is inserted at the left. This bit is also copied into the carry flag. The operation can be understood by imagining that the pipe used for shifting has been molded such that both ends coincide. Now when the first ball is forced to move forward, every ball moves one step forward with the last ball entering the pipe from its other end occupying the first ball's old position. The carry basket takes a snapshot of this ball leaving one end of the pipe and entering from the other.

- **…push function..**

The push operation copies its operand on the stack, When an item is pushed on a decrementing stack, the top of the stack is first decremented and the element is then copied into this space.

- **(CS, DS, SS, and ES)**

The code segment register, data segment register, stack segment register, and the extra segment register are special registers related to the Intel segmented memory model

- **INT 4…….**

INT 4, Arithmetic Overflow, change of sign bit .The overflow flag is set if the sign bit unexpectedly changes as a result of a mathematical or logical instruction. However the overflow flag signals a real overflow only if the numbers in question are treated as signed numbers. So this interrupt is not automatically generated but
as a result of a special instruction INTO (interrupt on overflow) if the overflow flag is set. Otherwise the INTO instruction behaves like a NOP (no operation).

- **CMPS**

CMPS subtracts the source location DS:SI from the destination location ES:DI. Source and Destination are unaffected. SI and DI are updated accordingly. CMPS compares two blocks of memory for equality or inequality of the block. It subtracts byte by byte or word by word. If used with a REPE or a REPNE prefix is repeats as long as the blocks are same or as long as they are different. For example it can be used for find a substring. A substring is a string that is contained in another string. For example "has" is contained in "Mary has a little lamp." Using CMPS we can do the operation of a complex loop in a single instruction. Only the REPE and REPNE prefixes are meaningful with this instruction.

Question No: 17 (www.vugoogle.com: 2)
Why is it necessary to provide the segment and offset address in case of FAR jump ?

Segment and offset must be given to a far jump. Because, sometimes we may need to go from one code segment to another, and near and short jumps cannot take us there. Far jump must be used and a two byte segment and a two byte offset are given to it. It loads CS with the segment part and IP with the offset part.

Question No: 18 (www.vugoogle.com: 2)
What's your understanding about Incrementing and Decrementing Stack?
Whenever an element is pushed on the stack SP is decremented by two and whenever an element is popped on the stack SP is incremented by two.

A decrementing stack moves from higher addresses to lower addresses as elements are added in it while an incrementing stack moves from lower addresses to higher addresses as elements are added.
As the 8088 stack works on word sized elements. Single bytes cannot be pushed or popped from the stack.

Question No: 19 (www.vugoogle.com: 2)
Number2:
IF DF=0 what its represent and IF DF=1 what its represent ?

The direction of movement is controlled with the Direction Flag (DF) in the flags register. If this flag is cleared DF=0, the direction is from lower addresses towards higher addresses and if this flag is set DF=1, the direction is from higher addresses to lower addresses. If DF is cleared, DF = 0 this is called the autoincrement mode of string instruction, and if DF is set, DF=1, this is called the autodecrement mode. There are two instructions to set and clear the direction flag.

Question No: 20 (www.vugoogle.com: 3)
What is the Difference between CALL and RET
The CALL instruction allows temporary diversion and therefore reusability of code.
The word return holds in its meaning that we are to return from where we came and need no explicit destination.
Therefore RET takes no arguments and transfers control back to the instruction following the CALL that took us in this subroutine.
Question No: 21 (www.vugoogle.com: 3)
Tell the Formula to scroll up the screen

```
rep movsw                       scroll up

scrollup: push bp
mov bp,sp
push ax
push cx
push si
push di
push es
push ds
mov ax, 80 ; load chars per row in ax
mul byte [bp+4]                 ; calculate source position
mov si, ax                      ; load source position in si
push si                         ; save position for later use
shl si, 1                       ; convert to byte offset
mov cx, 2000                    ; number of screen locations
sub cx, ax                      ; count of words to move
mov ax, 0xb800
```

```
mov es, ax                  ; point es to video base
mov ds, ax                  ; point ds to video base
xor di, di                  ; point di to top left column
cld                         ; set auto increment mode
rep movsw                   ; scroll up
mov ax, 0x0720              ; space in normal attribute
pop cx                      ; count of positions to clear
rep stosw                   ; clear the scrolled space
pop ds
pop es
pop di
pop si
pop cx
pop ax
pop bp
ret 2
```

Question No: 22 (www.vugoogle.com: 5)
Explain how extended shifting is performed

Using our basic shifting and rotation instructions we can effectively shift a 32bit number in memory word by word. We cannot shift the whole number at once since our architecture is limited to word operations. The algorithm we use consists of just two instructions and we name it extended shifting.

```
num1: dd 40000
shl word [num1], 1
rcl word [num1+2], 1
```

The DD directive reserves a 32bit space in memory; however the value we placed there will fit in 16bits. So we can safely shift the number left 16 times.
The least significant word is accessible at num1 and the most significant word is accessible at num1+2.
The two instructions are carefully crafted such that the first one shifts the lower word towards the left and the most significant bit of that word is dropped in carry. With the next instruction we push that dropped bit into the least significant bit of the next word effectively joining the two 16bit words.
The final carry after the second instruction will be the most significant bit of the higher word, which for this number will always be zero.

Question No: 23 (www.vugoogle.com: 5)
Write a subroutine to calculate the string length.?

```
subroutine to calculate the length of a string
; takes the segment and offset of a string as parameters
strlen: push bp
mov bp,sp
push es
push cx
push di
les di, [bp+4]                  ; point es:di to string
mov cx, 0xffff                  ; load maximum number in cx
xor al, al                  ; load a zero in al
repne scasb                 ; find zero in the string
mov ax, 0xffff                  ; load maximum number in ax
sub ax, cx                              ; find change in cx
```

```
dec ax                    ; exclude null from length
pop di
pop cx
pop es
pop bp
ret 4
```

Describe the push function

The push function copies the operand on to the stack. When an item is pushed on a decrement stack, top of the stack is first decrement and element is then copied into this space

How string instruction work on block of data

In a string instructions, block have a start and end. Instructions can work from the start toward to end and from the end towards start. They can work in both directions and the have to be allowed to work in both directions; otherwise some operations with overlapping blocks become not possible.

Why we need to clear the stack?

When the parameters by pushed for subroutine are waste after the subroutine return. They must be clear from the stack be caller and the callee.

Difference b/w REP and REPE and (REPNE)

The REP allow instructions to operate on a number of data elements in a one instructions, REPE repeats the instructions string instructions while the zero flash is set and REPNE repeat while not equal or repeat the instruction while the zero flag is not set

Explain all characteristics of SCAS instruction

1. SCAS instructions has variety of functions it compare a source of byte or word in register AL or AX with destination string address by ES:DI and updating flags.
2. It is used to locate equality or in-equality in a string
3. SCAS is bit different from the other instructions.
4. REPE and REPNE are used with this type of instructions
5. This is more like CMP instructions that it does subtraction of the operands.

DESCIRBE LOCAL VARIABLE

Local variables are created by STACK and these variables are needed when subroutine are in execution and not afterwards. They do not take place like global variables. They are temporary not permanent. Local variables are created when the subroutine is call. There meaning is in subroutine not outside. There most convenient place is stack. Special manipulation is needed for this task. Same base pointer can be used to access the local variables.

Why REP prefix is not generally use in LODS instructions

REP prefix use with LODS is not meaningful as only last value loaded will be remaining in register. It is use in loop paired with STOS

Define the NOT operator

Not operator inverts the bits of byte or word operand. This is single operand instructions. And invert the result into 1's complement form.

Differentiate b/w Intra Segment and Inter Segment:

Out of line procedures in the temporary division, the concept of the round about near calls are called Intra Segment. On the other hand far calls are called inter-segment class.

Explain the use of TEST instruction.

The test instruction is used for bit testing. BX holds the mask and in every next iteration it is shifting left, as our concerned bit is now the next bit.

Explain LES and LDS:

LES and LDS load a segment register and a general purpose register from two consecutive memory locations. LES loads ES while LDS loads DS. Instructions has two parameters, one is the general purpose register to be loaded and the other is the memory location from which to load these registers. There major application of these instructions is when a subroutine receives a segment offset pair as an argument and the pair is to be loaded in a segment and an offset register.

Describe MOVS and CMPS instructions

The MOVS instruction is a byte transfer or word from the source location D:SI to the destination. ES: DI and updated SI and DI to point to the next locations. It is use to moave block of memory. Dif plays important role in case of overlapping blocks And CMPS instructions subtracts the source location DS: SI from the destination location ES: DI. Source and the destination are unaffected.

Explain MUL instruction in both cases (i) if the source operand is byte (ii) if the source operand is a word

MUL performs un unsigned mulitiplication of the source operand and the accumulator. In the case of source operand is a byte, then it is multiply by the register AL and the double length result will be return into AH and AL. On the other side if the source operand is a word, then it is mulitiplied by the register AX, and the double-length result is returned in registers DX and X.

IF DF=0 what its represent and IF DF=1 what its represent?

We can control direction with Direction Flag (DF) in the flag Register. If the DF is value with it will be auto increment. and if the flag Register DF=1 it can set auto decrement.

Purpose of INT 1

Int 1 is Single step Interrupt, This interrupt is used in debugging with trap Flag. If the trap flag is set the single step interrupt is generated after every instructions.

write any two control instructions?……

These are instructions that control the program execution and flow by playing with the instruction pointer and altering its normal behavior to point to the next instruction. Some examples are:

```
cmp  ax, 0
jne  1234
```

We are changing the program flow to the instruction at 1234 address if the condition that we checked becomes true.

```
dec  cl              ; decrement bit count
 jnz  checkbit       ; repeat if bits left
```

q=18…RET instruction…

The word return holds in its meaning
that we are to return from where we came and need no explicit destination.
Therefore RET takes no arguments and transfers control back to the instruction following the CALL that took us in this subroutine.

ROR instructions….

In the rotate right operation every bit moves one position to the right and the bit dropped from the right is inserted at the left. This bit is also copied into the carry flag. The operation can be understood by imagining that the pipe used for shifting has been molded such that both ends coincide. Now when the first ball is forced to move forward, every ball moves one step forward with the last ball entering the pipe from its other end occupying the first ball's old position. The carry basket takes a snapshot of this ball leaving one end of the pipe and entering from the other.

…push function..

The push operation copies its operand on the stack,
When an item is pushed on a decrementing stack, the top of the stack is first decremented and the element is then copied into this space.

(CS, DS, SS, and ES)
The code segment register, data segment register, stack segment register, and the extra segment register are special registers related to the Intel segmented memory model

INT 4…….
INT 4, Arithmetic Overflow, change of sign bit
The overflow flag is set if the sign bit unexpectedly changes as a result of a mathematical or logical instruction. However the overflow flag signals a real overflow only if the numbers in question are treated as signed numbers. So this interrupt is not automatically generated but as a result of a special instruction INTO (interrupt on overflow) if the overflow flag is set. Otherwise the INTO instruction behaves like a NOP (no operation).

Calculate physical address..
Bx=0x0100
DS=0xFFF0
Ans:
100F0

CMPS

CMPS subtracts the source location DS:SI from the destination location ES:DI. Source and Destination are unaffected. SI and DI are updated accordingly. CMPS compares two blocks of memory for equality or inequality of the block. It subtracts byte by byte or word by word. If used with a REPE or a REPNE prefix is repeats as long as the blocks are same or as long as they are different. For example it can be used for find a substring. A substring is a string that is contained in another string. For example “has” is contained in “Mary has a little lamp.” Using CMPS we can do the operation of a complex loop in a single instruction. Only the REPE and REPNE prefixes are meaningful with this instruction.