

Name : Tazmeen Afroz

Roll No : 22P-9252

```
; a program to add three numbers using byte variables
[org 0x0100]

; initialize stuff
mov ax, 0          ; reset the accumulator
mov bx, 0          ; set the counter

outerloop:
    add ax, [num1 + bx]
    add bx, 2

    cmp bx, 20      ; sets ZF=0 when they are equal
    jne outerloop

mov [result], ax

mov ax, 0x4c00
int 0x21

; Intel Software Developer Manual - EFLAGS and Instructions (Page 435)

num1: dw 10, 20, 30, 40, 50, 10, 20, 30, 40, 50
result: dw 0
```

Code Explanation:

The given code is a program to add three numbers using byte variables.

First, the program initializes the accumulator (ax) and the counter (bx) to zero . This is done using the "mov" instruction.

Next, the program enters a loop labeled as "outerloop". In each iteration of the loop, the program adds the value at the memory location [num1 + bx] to the accumulator (ax) using the "add" instruction . The value at [num1 + bx] is accessed by adding the index bx to the base address of the num1 array.

After adding the value, the program increments the counter (bx) by 2 using the "add" instruction. This is done to move to the next index in the num1 array, as each element in the array is 2 bytes long .

The program then compares the value of bx with 20 using the "cmp" instruction. If bx and 20 are not equal, the zero flag (ZF) is set to 0. The "jne" instruction is a

conditional jump that checks the value of the zero flag. If the zero flag is not set (ZF=0), the program jumps back to the "outerloop" label and continues with the next iteration. This ensures that the loop continues until the value of bx is equal to 20.

Once the loop finishes, the program stores the final result in the memory location "result" using the "mov" instruction .

Conditional Jump Explanation:

The jne instruction stands for "jump if not equal." It is used to jump to a specified location in the code if the zero flag (ZF) is not set. The zero flag is set by certain instructions (like cmp) when the result of the comparison is zero (i.e., the two operands are equal). If the zero flag is not set, it means the operands are not equal, and the jne instruction causes the program to jump to the specified location.

Conditional Jump is Used in My Program:

In my program, the cmp instruction is used to compare the value of the bx register with 20. This comparison is done to determine if the loop has iterated over all the elements in the num1 array. The cmp instruction sets the zero flag if bx is equal to 20, and it clears the zero flag if bx is not equal to 20.

After the cmp instruction, the jne instruction checks the state of the zero flag. If the zero flag is not set (meaning bx is not equal to 20), the jne instruction causes the program to jump back to the outerloop label, and the loop continues with the next iteration. This ensures that the loop continues to add the numbers in the num1 array until all elements have been processed.

To provide a better understanding, I will now provide a screenshot with every step, including the changes in the values of ax and bx in each loop. :

| DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD | | | | | | | | | | | | | | | - | | X | | | | | | | | | | |
|--|---------|--------------|---------|----------|------|-------------------------|---------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-------------------|-------|--|--|--|
| AX 0000 | SI 0000 | CS 19F5 | IP 0106 | Stack +0 | 0000 | Flags 7200 | | | | | | | | | | | | | | | | | | | | | |
| BX 0000 | DI 0000 | DS 19F5 | | +2 | 20CD | | | | | | | | | | | | | | | | | | | | | | |
| CX 0032 | BP 0000 | ES 19F5 | HS 19F5 | +4 | 9FFF | OF DF IF SF ZF AF PF CF | | | | | | | | | | | | | | | | | | | | | |
| DX 0000 | SP FFFE | SS 19F5 | FS 19F5 | +6 | EA00 | 0 0 1 0 0 0 0 0 | | | | | | | | | | | | | | | | | | | | | |
| CMD > | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 000A | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0103 BB0000 | MOV | BX,0000 | | | | | DS:0000 | CD | 20 | FF | 9F | 00 | EA | F0 | FE | | | | | | | | | | | | |
| 0106 03871C01 | ADD | AX,[011C+BX] | | | | | DS:0008 | AD | DE | 1B | 05 | C5 | 06 | 00 | 00 | | | | | | | | | | | | |
| 010A 81C30200 | ADD | BX,0002 | | | | | DS:0010 | 18 | 01 | 10 | 01 | 18 | 01 | 92 | 01 | | | | | | | | | | | | |
| 010E 81FB1400 | CMP | BX,0014 | | | | | DS:0018 | 01 | 01 | 01 | 00 | 02 | FF | FF | FF | | | | | | | | | | | | |
| 0112 75F2 | JNZ | 0106 | | | | | DS:0020 | FF | FF | FF | FF | FF | FF | FF | FF | | | | | | | | | | | | |
| 0114 A33001 | MOV | [0130],AX | | | | | DS:0028 | FF | FF | FF | FF | EB | 19 | C0 | 11 | | | | | | | | | | | | |
| 0117 B8004C | MOV | AX,4C00 | | | | | DS:0030 | A2 | 01 | 14 | 00 | 18 | 00 | F5 | 19 | | | | | | | | | | | | |
| 011A CD21 | INT | 21 | | | | | DS:0038 | FF | FF | FF | FF | 00 | 00 | 00 | 00 | | | | | | | | | | | | |
| 011C 0A00 | OR | AL,[BX+SI] | | | | | DS:0040 | 05 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | | | | | | | | | | | | |
| | | | | | | | DS:0048 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | | | | | | | | | | | | |
| 2 | | | | | | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | | | | | |
| DS:0000 | | | | | | | CD | 20 | FF | 9F | 00 | EA | F0 | FE | AD | DE | 1B | 05 | C5 | 06 | 00 | 00 | = f.Ω≡ i ..+.... | | | | |
| DS:0010 | | | | | | | 18 | 01 | 10 | 01 | 18 | 01 | 92 | 01 | 01 | 01 | 00 | 02 | FF | FF | FF | FF |ff. | | | | |
| DS:0020 | | | | | | | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | EB | 19 | C0 | 11 | δ.L. | | | | |
| DS:0030 | | | | | | | A2 | 01 | 14 | 00 | 18 | 00 | F5 | 19 | FF | FF | FF | FF | 00 | 00 | 00 | 00 | ó.....J. | | | | |
| DS:0040 | | | | | | | 05 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | | | | |
| 1 Step 2ProcStep 3Retrieve 4Help ON 5BRK Menu 6 7 up 8 dn 9 le 10 ri | | | | | | | | | | | | | | | | | | | | | | | | | | | |

```

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD - X
AX 000A SI 0000 CS 19F5 IP 010E Stack +0 0000 Flags 7200
BX 0002 DI 0000 DS 19F5 +2 20CD
CX 0032 BP 0000 ES 19F5 HS 19F5 +4 9FFF OF DF IF SF ZF AF PF CF
DX 0000 SP FFFE SS 19F5 FS 19F5 +6 EA00 0 0 1 0 0 0 0 0

CMD >

010A 81C30200 ADD BX,0002
010E 81FB1400 CMP BX,0014
0112 75F2 JNZ 0106
0114 A33001 MOV [0130],AX
0117 B8004C MOV AX,4C00
011A CD21 INT 21
011C 0A00 OR AL,[BX+SI]
011E 1400 ADC AL,00
0120 1E PUSH DS

1
DS:0000 CD 20 FF 9F 00 EA F0 FE
DS:0008 AD DE 1B 05 C5 06 00 00
DS:0010 18 01 10 01 18 01 92 01
DS:0018 01 01 01 00 02 FF FF FF
DS:0020 FF FF FF FF FF FF FF FF
DS:0028 FF FF FF FF EB 19 C0 11
DS:0030 A2 01 14 00 18 00 F5 19
DS:0038 FF FF FF FF 00 00 00 00
DS:0040 05 00 00 00 00 00 00 00
DS:0048 00 00 00 00 00 00 00 00

2
DS:0000 CD 20 FF 9F 00 EA F0 FE AD DE 1B 05 C5 06 00 00 = f.Ω≡ i |..†...
DS:0010 18 01 10 01 18 01 92 01 01 01 01 00 02 FF FF FF .....ff. ....
DS:0020 FF FF FF FF FF FF FF FF FF FF FF FF EB 19 C0 11 δ.L.
DS:0030 A2 01 14 00 18 00 F5 19 FF FF FF FF 00 00 00 00 ó.....J. ....
DS:0040 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

1 Step 2ProcStep 3Retrieve 4Help ON 5BRK Menu 6 7 up 8 dn 9 le 10 ri

```

Explanation of the screenshot:

First Loop Iteration

- **add ax, [num1 + bx]:** Adds the value at the memory location [num1 + bx] to ax. Since bx is initially 0, this adds the first number 10 to ax.
- **add bx, 2:** Increments bx by 2. This is because each number in the array is stored as a word (2 bytes), so we need to skip over the next number in the array.
- **cmp bx, 20:** Compares bx to 20. Since bx is now 2, the comparison fails, and the loop continues

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD

| | | | | | |
|---------|---------|---------|---------|---------------|-------------------------|
| AX 001E | SI 0000 | CS 19F5 | IP 010E | Stack +0 0000 | Flags 7200 |
| BX 0004 | DI 0000 | DS 19F5 | | +2 20CD | |
| CX 0032 | BP 0000 | ES 19F5 | HS 19F5 | +4 9FFF | OF DF IF SF ZF AF PF CF |
| DX 0000 | SP FFFE | SS 19F5 | FS 19F5 | +6 EA00 | 0 0 1 0 0 0 0 0 |

CMD >

| | | | | | | | | | | | |
|---------------|------|------------|---------|----|----|----|----|----|----|----|----|
| 010A 81C30200 | ADD | BX,0002 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 010E 81FB1400 | CMP | BX,0014 | DS:0000 | CD | 20 | FF | 9F | 00 | EA | F0 | FE |
| 0112 75F2 | JNZ | 0106 | DS:0008 | AD | DE | 1B | 05 | C5 | 06 | 00 | 00 |
| 0114 A33001 | MOV | [0130],AX | DS:0010 | 18 | 01 | 10 | 01 | 18 | 01 | 92 | 01 |
| 0117 B8004C | MOV | AX,4C00 | DS:0018 | 01 | 01 | 01 | 00 | 02 | FF | FF | FF |
| 011A CD21 | INT | 21 | DS:0020 | FF | FF | FF | FF | FF | FF | FF | FF |
| 011C 0A00 | OR | AL,[BX+SI] | DS:0028 | FF | FF | FF | FF | EB | 19 | C0 | 11 |
| 011E 1400 | ADC | AL,00 | DS:0030 | A2 | 01 | 14 | 00 | 18 | 00 | F5 | 19 |
| 0120 1E | PUSH | DS | DS:0038 | FF | FF | FF | FF | 00 | 00 | 00 | 00 |
| | | | DS:0040 | 05 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| | | | DS:0048 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |

| | | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| DS:0000 | CD | 20 | FF | 9F | 00 | EA | F0 | FE | AD | DE | 1B | 05 | C5 | 06 | 00 | 00 |
| DS:0010 | 18 | 01 | 10 | 01 | 18 | 01 | 92 | 01 | 01 | 01 | 00 | 02 | FF | FF | FF | FF |
| DS:0020 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | EB | 19 | C0 | 11 | |
| DS:0030 | A2 | 01 | 14 | 00 | 18 | 00 | F5 | 19 | FF | FF | FF | FF | 00 | 00 | 00 | 00 |
| DS:0040 | 05 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |

= f.Ω≡■ ÷|..†...
.....ff.
δ.L.
ó.....J.
.....

1 Step 2ProcStep 3Retrieve 4Help ON 5BRK Menu 6 7 up 8 dn 9 le 10 ri

Explanation of the screenshot:

Second Loop Iteration

- **add ax, [num1 + bx]:** Adds the value at the memory location [num1 + bx] to ax. Since bx is now 2, this adds the second number 20 to ax. So, ax becomes 30 (10 + 20)
- **add bx, 2:** Increments bx by 2. Now bx is 4.
- **cmp bx, 20:** Compares bx to 20. Since bx is now 4, the comparison fails, and the loop continues.

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD

| | | | | | |
|---------|---------|---------|---------|---------------|-------------------------|
| AX 003C | SI 0000 | CS 19F5 | IP 010E | Stack +0 0000 | Flags 7204 |
| BX 0006 | DI 0000 | DS 19F5 | | +2 20CD | |
| CX 0032 | BP 0000 | ES 19F5 | HS 19F5 | +4 9FFF | OF DF IF SF ZF AF PF CF |
| DX 0000 | SP FFFE | SS 19F5 | FS 19F5 | +6 EA00 | 0 0 1 0 0 0 1 0 |

CMD >

| | | | | | | | | | | | |
|---------------|------|------------|---------|----|----|----|----|----|----|----|----|
| 010A 81C30200 | ADD | BX,0002 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 010E 81FB1400 | CMP | BX,0014 | DS:0000 | CD | 20 | FF | 9F | 00 | EA | F0 | FE |
| 0112 75F2 | JNZ | 0106 | DS:0008 | AD | DE | 1B | 05 | C5 | 06 | 00 | 00 |
| 0114 A33001 | MOV | [0130],AX | DS:0010 | 18 | 01 | 10 | 01 | 18 | 01 | 92 | 01 |
| 0117 B8004C | MOV | AX,4C00 | DS:0018 | 01 | 01 | 01 | 00 | 02 | FF | FF | FF |
| 011A CD21 | INT | 21 | DS:0020 | FF | FF | FF | FF | FF | FF | FF | FF |
| 011C 0A00 | OR | AL,[BX+SI] | DS:0028 | FF | FF | FF | FF | EB | 19 | C0 | 11 |
| 011E 1400 | ADC | AL,00 | DS:0030 | A2 | 01 | 14 | 00 | 18 | 00 | F5 | 19 |
| 0120 1E | PUSH | DS | DS:0038 | FF | FF | FF | FF | 00 | 00 | 00 | 00 |
| | | | DS:0040 | 05 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| | | | DS:0048 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |

| | | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| DS:0000 | CD | 20 | FF | 9F | 00 | EA | F0 | FE | AD | DE | 1B | 05 | C5 | 06 | 00 | 00 |
| DS:0010 | 18 | 01 | 10 | 01 | 18 | 01 | 92 | 01 | 01 | 01 | 00 | 02 | FF | FF | FF | FF |
| DS:0020 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | EB | 19 | C0 | 11 | |
| DS:0030 | A2 | 01 | 14 | 00 | 18 | 00 | F5 | 19 | FF | FF | FF | FF | 00 | 00 | 00 | 00 |
| DS:0040 | 05 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |

= f.Ω≡■ ÷|..†...
.....ff.
δ.L.
ó.....J.
.....

1 Step 2ProcStep 3Retrieve 4Help ON 5BRK Menu 6 7 up 8 dn 9 le 10 ri

Explanation of the screenshot:

Third Loop Iteration

- add ax, [num1 + bx]:** Adds the value at the memory location [num1 + bx] to ax. Since bx is now 4, this adds the third number 30 to ax. So, ax becomes 60 (30 + 30).
- add bx, 2:** Increments bx by 2. Now bx is 6.
- cmp bx, 20:** Compares bx to 20. Since bx is now 6, the comparison fails, and the loop continues.

```

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD - X
AX 0064 SI 0000 CS 19F5 IP 010E Stack +0 0000 Flags 7200
BX 0008 DI 0000 DS 19F5 +2 20CD
CX 0032 BP 0000 ES 19F5 HS 19F5 +4 9FFF OF DF IF SF ZF AF PF CF
DX 0000 SP FFFE SS 19F5 FS 19F5 +6 EA00 0 0 1 0 0 0 0 0

CMD >

010A 81C30200 ADD BX,0002
010E 81FB1400 CMP BX,0014
0112 75F2 JNZ 0106
0114 A33001 MOV [0130],AX
0117 B8004C MOV AX,4C00
011A CD21 INT 21
011C 0A00 OR AL,[BX+SI]
011E 1400 ADC AL,00
0120 1E PUSH DS

1
DS:0000 CD 20 FF 9F 00 EA F0 FE
DS:0008 AD DE 1B 05 C5 06 00 00
DS:0010 18 01 10 01 18 01 92 01
DS:0018 01 01 01 00 02 FF FF FF
DS:0020 FF FF FF FF FF FF FF FF
DS:0028 FF FF FF FF EB 19 C0 11
DS:0030 A2 01 14 00 18 00 F5 19
DS:0038 FF FF FF FF 00 00 00 00
DS:0040 05 00 00 00 00 00 00 00
DS:0048 00 00 00 00 00 00 00 00

2
0 1 2 3 4 5 6 7 8 9 A B C D E F
DS:0000 CD 20 FF 9F 00 EA F0 FE AD DE 1B 05 C5 06 00 00 = f.Ω≡ i |..†...
DS:0010 18 01 10 01 18 01 92 01 01 01 01 00 02 FF FF FF .....ff. ....
DS:0020 FF FF FF FF FF FF FF FF FF FF FF FF EB 19 C0 11 δ.L.
DS:0030 A2 01 14 00 18 00 F5 19 FF FF FF FF 00 00 00 00 ó.....J. ....
DS:0040 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

1 Step 2ProcStep 3Retrieve 4Help ON 5BRK Menu 6 7 up 8 dn 9 le 10 ri

```

Explanation of the screenshot:

Fourth Loop Iteration

- **add ax, [num1 + bx]:** Adds the value at the memory location [num1 + bx] to ax. Since bx is now 6, this adds the fourth number 40 to ax. So, ax becomes 100 (60 + 40).
- **add bx, 2:** Increments bx by 2. Now bx is 8.
- **cmp bx, 20:** Compares bx to 20. Since bx is now 8, the comparison fails, and the loop continues.

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD

| | | | | | |
|---------|---------|---------|---------|---------------|-------------------------|
| AX 0096 | SI 0000 | CS 19F5 | IP 010E | Stack +0 0000 | Flags 7204 |
| BX 000A | DI 0000 | DS 19F5 | | +2 20CD | |
| CX 0032 | BP 0000 | ES 19F5 | HS 19F5 | +4 9FFF | OF DF IF SF ZF AF PF CF |
| DX 0000 | SP FFFE | SS 19F5 | FS 19F5 | +6 EA00 | 0 0 1 0 0 0 1 0 |

CMD >

| | | | | | | | | | | | |
|---------------|------|------------|---------|----|----|----|----|----|----|----|----|
| 010A 81C30200 | ADD | BX,0002 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 010E 81FB1400 | CMP | BX,0014 | DS:0000 | CD | 20 | FF | 9F | 00 | EA | F0 | FE |
| 0112 75F2 | JNZ | 0106 | DS:0008 | AD | DE | 1B | 05 | C5 | 06 | 00 | 00 |
| 0114 A33001 | MOV | [0130],AX | DS:0010 | 18 | 01 | 10 | 01 | 18 | 01 | 92 | 01 |
| 0117 B8004C | MOV | AX,4C00 | DS:0018 | 01 | 01 | 01 | 00 | 02 | FF | FF | FF |
| 011A CD21 | INT | 21 | DS:0020 | FF | FF | FF | FF | FF | FF | FF | FF |
| 011C 0A00 | OR | AL,[BX+SI] | DS:0028 | FF | FF | FF | FF | EB | 19 | C0 | 11 |
| 011E 1400 | ADC | AL,00 | DS:0030 | A2 | 01 | 14 | 00 | 18 | 00 | F5 | 19 |
| 0120 1E | PUSH | DS | DS:0038 | FF | FF | FF | FF | 00 | 00 | 00 | 00 |
| | | | DS:0040 | 05 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| | | | DS:0048 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |

| | | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| DS:0000 | CD | 20 | FF | 9F | 00 | EA | F0 | FE | AD | DE | 1B | 05 | C5 | 06 | 00 | 00 |
| DS:0010 | 18 | 01 | 10 | 01 | 18 | 01 | 92 | 01 | 01 | 01 | 00 | 02 | FF | FF | FF | FF |
| DS:0020 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | EB | 19 | C0 | 11 | |
| DS:0030 | A2 | 01 | 14 | 00 | 18 | 00 | F5 | 19 | FF | FF | FF | FF | 00 | 00 | 00 | 00 |
| DS:0040 | 05 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |

= f.Ω≡ i |..†...
.....ff.
δ.L.
ó.....J.
.....

1 Step 2ProcStep 3Retrieve 4Help ON 5BRK Menu 6 7 up 8 dn 9 le 10 ri

Explanation of the screenshot:

Fifth Loop Iteration

- add ax, [num1 + bx]:** Adds the value at the memory location [num1 + bx] to ax. Since bx is now 8, this adds the fifth number in the array 50 to ax. So, ax becomes 150 (100 + 50).
- add bx, 2:** Increments bx by 2. Now bx is 10.
- cmp bx, 20:** Compares bx to 20. Since bx is now 10, the comparison fails, and the loop continues.

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD

| | | | | | |
|---------|---------|---------|---------|---------------|-------------------------|
| AX 00A0 | SI 0000 | CS 19F5 | IP 010E | Stack +0 0000 | Flags 7204 |
| BX 000C | DI 0000 | DS 19F5 | | +2 20CD | |
| CX 0032 | BP 0000 | ES 19F5 | HS 19F5 | +4 9FFF | OF DF IF SF ZF AF PF CF |
| DX 0000 | SP FFFE | SS 19F5 | FS 19F5 | +6 EA00 | 0 0 1 0 0 0 1 0 |

CMD >

| | | | | | | | | | | | |
|---------------|------|------------|---------|----|----|----|----|----|----|----|----|
| 010A 81C30200 | ADD | BX,0002 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 010E 81FB1400 | CMP | BX,0014 | DS:0000 | CD | 20 | FF | 9F | 00 | EA | F0 | FE |
| 0112 75F2 | JNZ | 0106 | DS:0008 | AD | DE | 1B | 05 | C5 | 06 | 00 | 00 |
| 0114 A33001 | MOV | [0130],AX | DS:0010 | 18 | 01 | 10 | 01 | 18 | 01 | 92 | 01 |
| 0117 B8004C | MOV | AX,4C00 | DS:0018 | 01 | 01 | 01 | 00 | 02 | FF | FF | FF |
| 011A CD21 | INT | 21 | DS:0020 | FF | FF | FF | FF | FF | FF | FF | FF |
| 011C 0A00 | OR | AL,[BX+SI] | DS:0028 | FF | FF | FF | FF | EB | 19 | C0 | 11 |
| 011E 1400 | ADC | AL,00 | DS:0030 | A2 | 01 | 14 | 00 | 18 | 00 | F5 | 19 |
| 0120 1E | PUSH | DS | DS:0038 | FF | FF | FF | FF | 00 | 00 | 00 | 00 |
| | | | DS:0040 | 05 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| | | | DS:0048 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |

2

| | | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| DS:0000 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| DS:0000 | CD | 20 | FF | 9F | 00 | EA | F0 | FE | AD | DE | 1B | 05 | C5 | 06 | 00 | 00 |
| DS:0010 | 18 | 01 | 10 | 01 | 18 | 01 | 92 | 01 | 01 | 01 | 00 | 02 | FF | FF | FF | FF |
| DS:0020 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | EB | 19 | C0 | 11 | |
| DS:0030 | A2 | 01 | 14 | 00 | 18 | 00 | F5 | 19 | FF | FF | FF | FF | 00 | 00 | 00 | 00 |
| DS:0040 | 05 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |

= f.Ω≡■ ÷|..†...
.....ff.
δ.L.
ó.....J.
.....

1 Step 2ProcStep 3Retrieve 4Help ON 5BRK Menu 6 7 up 8 dn 9 le 10 ri

Explanation of the screenshot:

Sixth Loop Iteration

- add ax, [num1 + bx]:** Adds the value at the memory location [num1 + bx] to ax. Since bx is now 10, this adds the sixth number 10 to ax. So, ax becomes 160 (150 + 10).
- add bx, 2:** Increments bx by 2. Now bx is 12.
- cmp bx, 20:** Compares bx to 20. Since bx is now 12, the comparison fails, and the loop continues.

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD

| | | | | | |
|---------|---------|---------|---------|---------------|-------------------------|
| AX 00B4 | SI 0000 | CS 19F5 | IP 010E | Stack +0 0000 | Flags 7200 |
| BX 000E | DI 0000 | DS 19F5 | | +2 20CD | |
| CX 0032 | BP 0000 | ES 19F5 | HS 19F5 | +4 9FFF | OF DF IF SF ZF AF PF CF |
| DX 0000 | SP FFFE | SS 19F5 | FS 19F5 | +6 EA00 | 0 0 1 0 0 0 0 0 |

CMD >

| | | | | | | | | | | | |
|---------------|------|------------|---------|----|----|----|----|----|----|----|----|
| 010A 81C30200 | ADD | BX,0002 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 010E 81FB1400 | CMP | BX,0014 | DS:0000 | CD | 20 | FF | 9F | 00 | EA | F0 | FE |
| 0112 75F2 | JNZ | 0106 | DS:0008 | AD | DE | 1B | 05 | C5 | 06 | 00 | 00 |
| 0114 A33001 | MOV | [0130],AX | DS:0010 | 18 | 01 | 10 | 01 | 18 | 01 | 92 | 01 |
| 0117 B8004C | MOV | AX,4C00 | DS:0018 | 01 | 01 | 01 | 00 | 02 | FF | FF | FF |
| 011A CD21 | INT | 21 | DS:0020 | FF | FF | FF | FF | FF | FF | FF | FF |
| 011C 0A00 | OR | AL,[BX+SI] | DS:0028 | FF | FF | FF | FF | EB | 19 | C0 | 11 |
| 011E 1400 | ADC | AL,00 | DS:0030 | A2 | 01 | 14 | 00 | 18 | 00 | F5 | 19 |
| 0120 1E | PUSH | DS | DS:0038 | FF | FF | FF | FF | 00 | 00 | 00 | 00 |
| | | | DS:0040 | 05 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| | | | DS:0048 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |

2

| | | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| DS:0000 | CD | 20 | FF | 9F | 00 | EA | F0 | FE | AD | DE | 1B | 05 | C5 | 06 | 00 | 00 |
| DS:0010 | 18 | 01 | 10 | 01 | 18 | 01 | 92 | 01 | 01 | 01 | 00 | 02 | FF | FF | FF | FF |
| DS:0020 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | EB | 19 | C0 | 11 | |
| DS:0030 | A2 | 01 | 14 | 00 | 18 | 00 | F5 | 19 | FF | FF | FF | FF | 00 | 00 | 00 | 00 |
| DS:0040 | 05 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |

= f.Ω≡ i |..†...
.....ff.
δ.L.
ó.....J.
.....

1 Step 2ProcStep 3Retrieve 4Help ON 5BRK Menu 6 7 up 8 dn 9 le 10 ri

Explanation of the screenshot:

Seventh Loop Iteration

- add ax, [num1 + bx]:** Adds the value at the memory location [num1 + bx] to ax. Since bx is now 12, this adds the seventh number 20 to ax. So, ax becomes 180 (160 + 20).
- add bx, 2:** Increments bx by 2. Now bx is 14.
- cmp bx, 20:** Compares bx to 20. Since bx is now 14, the comparison fails, and the loop continues.

```

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD
AX 00D2 SI 0000 CS 19F5 IP 0112 Stack +0 0000 Flags 7295
BX 0010 DI 0000 DS 19F5 +2 20CD
CX 0032 BP 0000 ES 19F5 HS 19F5 +4 9FFF OF DF IF SF ZF AF PF CF
DX 0000 SP FFFE SS 19F5 FS 19F5 +6 EA00 0 0 1 1 0 1 1 1

CMD >
010E 81FB1400 CMP BX,0014
0112 75F2 JNZ 0106
0114 A33001 MOV [0130],AX
0117 B8004C MOV AX,4C00
011A CD21 INT 21
011C 0A00 OR AL,[BX+SI]
011E 1400 ADC AL,00
0120 1E PUSH DS
0121 0028 ADD [BX+SI],CH

1
DS:0000 CD 20 FF 9F 00 EA F0 FE
DS:0008 AD DE 1B 05 C5 06 00 00
DS:0010 18 01 10 01 18 01 92 01
DS:0018 01 01 01 00 02 FF FF FF
DS:0020 FF FF FF FF FF FF FF FF
DS:0028 FF FF FF FF EB 19 C0 11
DS:0030 A2 01 14 00 18 00 F5 19
DS:0038 FF FF FF FF 00 00 00 00
DS:0040 05 00 00 00 00 00 00 00
DS:0048 00 00 00 00 00 00 00 00

2
0 1 2 3 4 5 6 7 8 9 A B C D E F
DS:0000 CD 20 FF 9F 00 EA F0 FE AD DE 1B 05 C5 06 00 00 = f.Ω≡■ ÷|..†...
DS:0010 18 01 10 01 18 01 92 01 01 01 01 00 02 FF FF FF .....ff. ....
DS:0020 FF FF FF FF FF FF FF FF FF FF FF FF EB 19 C0 11 δ.L.
DS:0030 A2 01 14 00 18 00 F5 19 FF FF FF FF 00 00 00 00 ó.....J. ....
DS:0040 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

1 Step 2ProcStep 3Retrieve 4Help ON 5BRK Menu 6 7 up 8 dn 9 le 10 ri

```

Explanation of the screenshot:

Eighth Loop Iteration

- **add ax, [num1 + bx]:** Adds the value at the memory location [num1 + bx] to ax. Since bx is now 14, this adds the eighth number 30 to ax. So, ax becomes 210 (180 + 30).
- **add bx, 2:** Increments bx by 2. Now bx is 16.
- **cmp bx, 20:** Compares bx to 20. Since bx is now 16, the comparison fails, and the loop continues.

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD

| | | | | | |
|---------|---------|---------|---------|---------------|-------------------------|
| AX 00FA | SI 0000 | CS 19F5 | IP 010E | Stack +0 0000 | Flags 7204 |
| BX 0012 | DI 0000 | DS 19F5 | | +2 20CD | |
| CX 0032 | BP 0000 | ES 19F5 | HS 19F5 | +4 9FFF | OF DF IF SF ZF AF PF CF |
| DX 0000 | SP FFFE | SS 19F5 | FS 19F5 | +6 EA00 | 0 0 1 0 0 0 1 0 |

CMD >

| | | | | | | | | | | | |
|---------------|------|------------|---------|----|----|----|----|----|----|----|----|
| 010A 81C30200 | ADD | BX,0002 | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 010E 81FB1400 | CMP | BX,0014 | DS:0000 | CD | 20 | FF | 9F | 00 | EA | F0 | FE |
| 0112 75F2 | JNZ | 0106 | DS:0008 | AD | DE | 1B | 05 | C5 | 06 | 00 | 00 |
| 0114 A33001 | MOV | [0130],AX | DS:0010 | 18 | 01 | 10 | 01 | 18 | 01 | 92 | 01 |
| 0117 B8004C | MOV | AX,4C00 | DS:0018 | 01 | 01 | 01 | 00 | 02 | FF | FF | FF |
| 011A CD21 | INT | 21 | DS:0020 | FF | FF | FF | FF | FF | FF | FF | FF |
| 011C 0A00 | OR | AL,[BX+SI] | DS:0028 | FF | FF | FF | FF | EB | 19 | C0 | 11 |
| 011E 1400 | ADC | AL,00 | DS:0030 | A2 | 01 | 14 | 00 | 18 | 00 | F5 | 19 |
| 0120 1E | PUSH | DS | DS:0038 | FF | FF | FF | FF | 00 | 00 | 00 | 00 |
| | | | DS:0040 | 05 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| | | | DS:0048 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |

| | | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| DS:0000 | CD | 20 | FF | 9F | 00 | EA | F0 | FE | AD | DE | 1B | 05 | C5 | 06 | 00 | 00 |
| DS:0010 | 18 | 01 | 10 | 01 | 18 | 01 | 92 | 01 | 01 | 01 | 00 | 02 | FF | FF | FF | FF |
| DS:0020 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | EB | 19 | C0 | 11 | |
| DS:0030 | A2 | 01 | 14 | 00 | 18 | 00 | F5 | 19 | FF | FF | FF | FF | 00 | 00 | 00 | 00 |
| DS:0040 | 05 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |

= f.Ω≡■ ÷|..†...
.....ff.
δ.L.
ó.....J.
.....

1 Step 2ProcStep 3Retrieve 4Help ON 5BRK Menu 6 7 up 8 dn 9 le 10 ri

Explanation of the screenshot:

Ninth Loop Iteration

- add ax, [num1 + bx]:** Adds the value at the memory location [num1 + bx] to ax. Since bx is now 16, this adds the ninth number 40 to ax. So, ax becomes 250(210+ 40).
- add bx, 2:** Increments bx by 2. Now bx is 18.
- cmp bx, 20:** Compares bx to 20. Since bx is now 18, the comparison fails, and the loop continues.

```

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD
AX 012C SI 0000 CS 19F5 IP 010E Stack +0 0000 Flags 7204
BX 0014 DI 0000 DS 19F5 +2 20CD
CX 0032 BP 0000 ES 19F5 HS 19F5 +4 9FFF OF DF IF SF ZF AF PF CF
DX 0000 SP FFFE SS 19F5 FS 19F5 +6 EA00 0 0 1 0 0 0 1 0

CMD >

010A 81C30200 ADD BX,0002
010E 81FB1400 CMP BX,0014
0112 75F2 JNZ 0106
0114 A33001 MOV [0130],AX
0117 B8004C MOV AX,4C00
011A CD21 INT 21
011C 0A00 OR AL,[BX+SI]
011E 1400 ADC AL,00
0120 1E PUSH DS

1 0 1 2 3 4 5 6 7
DS:0000 CD 20 FF 9F 00 EA F0 FE
DS:0008 AD DE 1B 05 C5 06 00 00
DS:0010 18 01 10 01 18 01 92 01
DS:0018 01 01 01 00 02 FF FF FF
DS:0020 FF FF FF FF FF FF FF FF
DS:0028 FF FF FF FF EB 19 C0 11
DS:0030 A2 01 14 00 18 00 F5 19
DS:0038 FF FF FF FF 00 00 00 00
DS:0040 05 00 00 00 00 00 00 00
DS:0048 00 00 00 00 00 00 00 00

2 0 1 2 3 4 5 6 7 8 9 A B C D E F
DS:0000 CD 20 FF 9F 00 EA F0 FE AD DE 1B 05 C5 06 00 00 = f.Ω≡ i |..†...
DS:0010 18 01 10 01 18 01 92 01 01 01 01 00 02 FF FF FF .....ft. ....
DS:0020 FF FF FF FF FF FF FF FF FF FF FF FF FF EB 19 C0 11 δ.L.
DS:0030 A2 01 14 00 18 00 F5 19 FF FF FF FF 00 00 00 00 ó.....J. ....
DS:0040 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

1 Step 2ProcStep 3Retrieve 4Help ON 5BRK Menu 6 7 up 8 dn 9 le 10 ri

```

Explanation of the screenshot:

Tenth Loop Iteration

- **add ax, [num1 + bx]:** Adds the value at the memory location [num1 + bx] to ax. Since bx is now 18, this adds the tenth number 50 to ax. So, ax becomes 300 (250+ 50).
- **add bx, 2:** Increments bx by 2. Now bx is 20.
- **cmp bx, 20:** Compares bx to 20. Since bx is now 20, the comparison succeeds, and the loop terminates.

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD

| | | | | | |
|---------|---------|---------|---------|---------------|-------------------------|
| AX 012C | SI 0000 | CS 19F5 | IP 0117 | Stack +0 0000 | Flags 7244 |
| BX 0014 | DI 0000 | DS 19F5 | | +2 20CD | |
| CX 0032 | BP 0000 | ES 19F5 | HS 19F5 | +4 9FFF | OF DF IF SF ZF AF PF CF |
| DX 0000 | SP FFFE | SS 19F5 | FS 19F5 | +6 EA00 | 0 0 1 0 1 0 1 0 |

CMD >

| | | | | | | | | | | | |
|-------------|------|------------|---------|----|----|----|----|----|----|----|----|
| 0114 A33001 | MOV | [0130],AX | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0117 B8004C | MOV | AX,4C00 | DS:0000 | CD | 20 | FF | 9F | 00 | EA | F0 | FE |
| 011A CD21 | INT | 21 | DS:0008 | AD | DE | 1B | 05 | C5 | 06 | 00 | 00 |
| 011C 0A00 | OR | AL,[BX+SI] | DS:0010 | 18 | 01 | 10 | 01 | 18 | 01 | 92 | 01 |
| 011E 1400 | ADC | AL,00 | DS:0018 | 01 | 01 | 01 | 00 | 02 | FF | FF | FF |
| 0120 1E | PUSH | DS | DS:0020 | FF | FF | FF | FF | FF | FF | FF | FF |
| 0121 0028 | ADD | [BX+SI],CH | DS:0028 | FF | FF | FF | FF | EB | 19 | C0 | 11 |
| 0123 0032 | ADD | [BP+SI],DH | DS:0030 | A2 | 01 | 14 | 00 | 18 | 00 | F5 | 19 |
| 0125 000A | ADD | [BP+SI],CL | DS:0038 | FF | FF | FF | FF | 00 | 00 | 00 | 00 |
| | | | DS:0040 | 05 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| | | | DS:0048 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |

| | | | | | | | | | | | | | | | | |
|---------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| DS:0000 | CD | 20 | FF | 9F | 00 | EA | F0 | FE | AD | DE | 1B | 05 | C5 | 06 | 00 | 00 |
| DS:0010 | 18 | 01 | 10 | 01 | 18 | 01 | 92 | 01 | 01 | 01 | 00 | 02 | FF | FF | FF | FF |
| DS:0020 | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | EB | 19 | C0 | 11 | |
| DS:0030 | A2 | 01 | 14 | 00 | 18 | 00 | F5 | 19 | FF | FF | FF | FF | 00 | 00 | 00 | 00 |
| DS:0040 | 05 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |

= f.Ω≡■ ÷|..†...
.....ff.
δ.L.
ó.....J.
.....

1 Step 2ProcStep 3Retrieve 4Help ON 5BRK Menu 6 7 up 8 dn 9 le 10 ri

Finalization

- `mov [result], ax`: Stores the final sum in the result variable.
- `mov ax, 0x4c00`: Prepares to terminate the program.
- `int 0x21`: Calls the DOS interrupt to terminate the program.

```

[org 0x0100]

jmp start    ; see next instructions when you haven't yet executed this!

num1: dw 10, 20, 30, 40, 50, 10, 20, 30, 40, 50
result: dw 0

start:
; initialize stuff
mov ax, 0    ; reset the accumulator
mov bx, 0    ; set the counter

outerloop:
add ax, [num1 + bx]
add bx, 2

cmp bx, 20   ; sets ZF=0 when they are equal
jne outerloop

mov [result], ax

mov ax, 0x4c00
int 0x21

```

Unconditional Jump:

The unconditional jump (jmp) instruction in assembly language is used to transfer control to a different part of the program without any condition. It simply jumps to the specified location in the code and continues execution from there.

Unconditional Jump In My Program:

In my program, the jmp start instruction is placed at the very beginning, right after the org 0x0100 directive. This directive sets the origin, or starting address, of the program in memory. The jmp start instruction then immediately transfers control to the start label, effectively skipping over the num1 and result data declarations.

After the jmp start instruction, the program initializes the ax and bx registers, enters a loop to add the numbers stored in the num1 array, and finally stores the result in the result variable. The loop continues until the bx register is equal to 20, indicating that all elements in the num1 array have been processed.

Data Stored:

Despite the unconditional jump (jmp start) skipping the num1 and result data declarations, the data is still stored in memory.

Why:

When the assembler processes the program, it allocates space in the program's data segment for each variable declared in the program.

Where Data is Stored:

The screenshot shows the DOSBox 0.74-3 interface. At the top, it displays 'DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD'. Below this, a table of CPU registers is shown: AX 0000, SI 0000, CS 19F5, IP 0100, Stack +0 0000, Flags 7202; BX 0000, DI 0000, DS 19F5, +2 20CD; CX 0035, BP 0000, ES 19F5, HS 19F5, +4 9FFF, OF DF IF SF ZF AF PF CF; DX 0000, SP FFFE, SS 19F5, FS 19F5, +6 EA00, 0 0 1 0 0 0 0 0. Below the registers, the command line 'CMD >' is shown. The main window displays a list of instructions and their addresses: 0100 E91600 JMP 0119; 0103 0A00 OR AL, [BX+SI]; 0105 1400 ADC AL, 00; 0107 1E PUSH DS; 0108 0028 ADD [BX+SI], CH; 010A 0032 ADD [BP+SI], DH; 010C 000A ADD [BP+SI], CL; 010E 0014 ADD [SI], DL. To the right of the instructions, a memory dump is shown, starting at address 0100. The dump shows the following values: 0100: E9 16 00 0A 00 14 00 1E; 0108: 00 28 00 32 00 0A 00 14; 0110: 00 1E 00 28 00 32 00 00; 0118: 00 B8 00 00 B8 00 00 03; 0120: 87 03 01 81 C3 02 00 81; 0128: FB 14 00 75 F2 A3 17 01; 0130: B8 00 4C CD 21 D2 75 04; 0138: 85 C0 74 1C C7 46 DC 00; 0140: 00 8E 5E FC 83 7D 0E 00; 0148: 74 09 8B 46 F2 48 3B 46. At the bottom, a status bar shows '1 Step 2ProcStep 3Retrieve 4Help ON 5BRK Menu 6 7 up 8 dn 9 le 10 ri'.

In the above screenshot, the data is stored starting from address 0103. This indicates that the jump instruction itself occupies 3 bytes in memory, with its opcode being E9 16 00.