

1) Being aware of the c02-01.asm

- First, use the previous commands we read in lab to run the file c02-01.asm into the dosbox. Following the file's execution, the Dosbox emulator interface appears on our screen. At this point, we evaluate any modifications made from our earlier codes and highlight them.
- To see the code and the modifications from the preceding code, we first open c02-01.asm.

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

mov ax, [num1]          ; load first number in ax
; mov [num1], [num2]    ; illegal
mov bx, [num2]
add ax, bx
mov bx, [num3]
add ax, bx
mov [num4], ax
mov ax, 0x4c00
int 0x21

num1: dw 5
num2: dw 10
num3: dw 15
num4: dw 0
```

- This code demonstrates how we move the value in Ax using square brackets; this technique is known as sending value utilising the labels. The value of num1 in the brackets is derived from the label at the conclusion of the code, which indicates that the value is 5.
- Now test this code into the debugger and check the changes.

```

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD
AX 0000 SI 0000 CS 19F5 IP 0100 Stack +0 0000 Flags 7202
BX 0000 DI 0000 DS 19F5 +2 20CD
CX 001F 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 >

0100 A11701 MOV AX,[0117]
0103 8B1E1901 MOV BX,[0119]
0107 01D8 ADD AX,BX
0109 8B1E1B01 MOV BX,[011B]
010D 01D8 ADD AX,BX
010F A31D01 MOV [011D],AX
0112 B8004C MOV AX,4C00
0115 CD21 INT 21

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

```

- As we can see, there is an object in the brackets that is the num1 value's address. It takes that value from the end of the address, which is stored in memory, places it there (between the braces), and then moves it to the Axe register.

*That's the only thing that separates the new one from our earlier ones. Instead of using the addressing approach like we did in the past, we shift direct value into the Axe.

- Now, we can observe how much memory this code uses. Count starting from the right side and confirm using the file's attributes.

```

2 0 1 2 3 4 5 6 7 8 9 A B C D E F
DS:0100 A1 17 01 8B 1E 19 01 01 D8 8B 1E 1B 01 01 D8 A3
DS:0110 1D 01 B8 00 4C CD 21 05 00 0A 00 0F 00 00 00
DS:0120 F6 00 00 8B 46 F6 D1 E0 D1 E0 C5 5E D8 01 C3 8B
DS:0130 07 8B 57 02 85 D2 75 04 85 C0 74 1C C7 46 DC 00
DS:0140 00 8E 5E FC 83 7D 0E 00 74 09 8B 46 F2 48 3B 46

```

- same 31 bytes this code consumes in the memory, count that values from 0100 to selected region they equal to 31 also.

2) Being aware of the c02-02.asm:

- Let's open the code file now and start by identifying the differences.

Thus, we can observe the modification in the final lines where the values for num1 were given in the earlier code.

```
; a program to add three numbers accessed using a single label
[org 0x0100]

    mov ax, [num1]
    mov bx, [num1 + 2]    ; notice how we can do arithmetic here
    add ax, bx           ; also, why +2 and not +1?
    mov bx, [num1 + 4]
    add ax, bx
    mov [num1 + 6], ax    ; store sum at num1+6
    mov ax, 0x4c00
    int 0x21

num1:    dw 5
         dw 10
         dw 15
         dw 0
```

As you can see, we used to use three or four variables for this, but now we only use one variable that has a range of values, functioning similarly to an array.

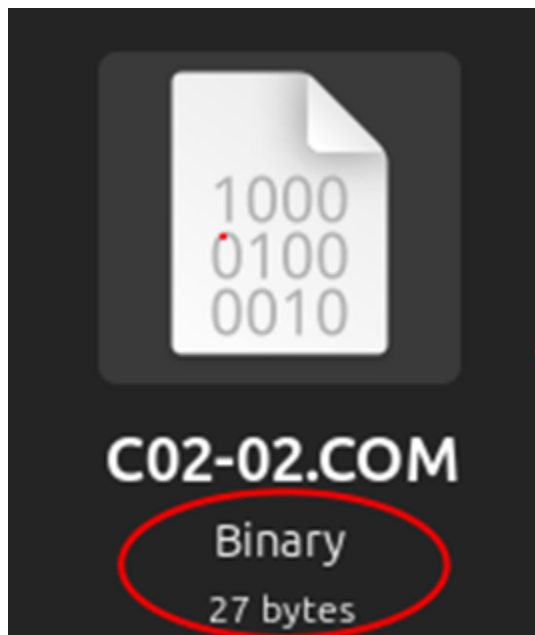
• However, the difficulty is how to feed numbers like 10, 15, thus using the same technique, we do it in an array, making leaps based on the bit widths. As you can see, the word "dw" is written with values that define it. We should be aware that this word requires two bytes of memory. Therefore, we always leap by two bytes for the subsequent value. Launch it in the debugger to verify.

Since dw was also used in the earlier code and I changed it to db, you can see that there is no difference.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
:0100	A1	17	01	8B	1E	19	01	01	D8	8B	1E	1B	01	01	D8	A3
:0110	1D	01	B8	00	4C	CD	21	05	0A	0F	00					
:0120	F6	00	00	8B	46	F6	D1	E0	D1	E0	C5	5E	D8	01	C3	8B
:0130	07	8B	57	02	85	D2	75	04	85	C0	74	1C	C7	46	DC	00
:0140	00	8E	5E	FC	83	7D	0E	00	74	09	8B	46	F2	48	3B	46

- Observe the distinction The values are now set to 1 bit. Naturally, this will have an impact on memory capacity, resulting in 27 bytes instead of 31 bytes. Let's see

"Observe the dimensions"



3) Understanding the c02-03.asm:

- Again lets check the new file

```
; a program to add three numbers accessed  
[org 0x0100]
```

```
mov ax, [num1]  
mov bx, [num1 + 2]  
add ax, bx  
mov bx, [num1 + 4]  
add ax, bx  
mov [num1 + 6], ax  
mov ax, 0x4c00  
int 0x21
```

```
num1: dw 5, 10, 15, 0
```

- As of right now, there is a slight variation in that every value is of the same data type. Thus, we may quickly retrieve the appropriate values.
- Launch the dosbox debugger and examine it.

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

AX 0000	SI 0000	CS 19F5	IP 0100	Stack +0 0000	Flags 7202
BX 0000	DI 0000	DS 19F5		+2 20CD	
CX 001F	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 > taking 4-bytes for each value

0100 A11701	MOV	AX, [0117]
0103 8B1E1901	MOV	BX, [0119]
0107 01D8	ADD	AX, BX
0109 8B1E1B01	MOV	BX, [011B]
010D 01D8	ADD	AX, BX
010F A31D01	MOV	[011D], AX
0112 B8004C	MOV	AX, 4C00
0115 CD21	INT	21

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 01 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

DS:0100	A1 17 01 8B 1E 19 01 01	D8 8B 1E 1B 01 01 D8 A3	í...ï....	†ï....†ú
DS:0110	1D 01 B8 00 4C CD 21 05	00 0A 00 0F 00 00 00 46	..ŕ.L=?.F
DS:0120	F6 00 00 8B 46 F6 D1 E0	D1 E0 C5 5E D8 01 C3 8B	÷...ïF÷ŕα	ŕα†^†.†ï
DS:0130	07 8B 57 02 85 D2 75 04	85 C0 74 1C C7 46 DC 00	..ïW.àŕu.	à^t.ŕF■.
DS:0140	00 8E 5E FC 83 7D 0E 00	74 09 8B 46 F2 48 3B 46	..Ä^"â}..	t.ïF≥H;F

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

- As a result, dw causes the stored values in memory to now require two bytes.

4) Being Aware of the c02-04.asm

- Open that file first, then look for any changes from the preceding one's code.

```

[org 0x0100]

    mov ax, [num1]
    mov [num1 + 6], ax    ; add this value to result

    mov ax, [num1 + 2]
    add [num1 + 6], ax

    mov ax, [num1 + 4]
    add [num1+6], ax

    mov ax, 0x4c00
    int 0x21

num1:    dw 5, 10, 15, 0

; should have the result separate!
; let's change that!

```

Therefore, instead of saving a trash value in this file, we choose to save a value of zero for the addition's result.

As you can see, we moved the value of ax into [num1+6], which implies at the zero location, in the second line. We add [num + 6] because we are using the word type to describe our label and we need to leap three values. Since every value comes after two bytes, we know that $2(\text{bytes}) * 3(\text{values to jump}) = 6$.

- Let's now try this in the debugger.

- After pressing F1
- See changing in value of Ax,also in IP register,i mean to say some general changes which we see in every code. Again...

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD																-		x		
AX	000A	SI	0000	CS	19F5	IP	0109	Stack	+0	0000	Flags	7200								
BX	0000	DI	0000	DS	19F5				+2	20CD										
CX	0000	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 >										0005										
0106 A11B01 MOV AX,[011B]										1 0 1 2 3 4 5 6 7										
0109 01061F01 ADD [011F],AX										DS:0000 CD 20 FF 9F 00 EA FF FF										
010D A11D01 MOV AX,[011D]										DS:0008 AD DE 1B 05 C5 06 00 00										
0110 01061F01 ADD [011F],AX										DS:0010 18 01 10 01 18 01 92 01										
0114 B8004C MOV AX,4C00										DS:0018 01 01 01 00 02 FF FF FF										
0117 CD21 INT 21										DS:0020 FF FF FF FF FF FF FF FF										
0119 05000A ADD AX,0A00										DS:0028 FF FF FF FF EB 19 E6 11										
011C 000F ADD [BX],CL										DS:0030 A2 01 14 00 18 00 F5 19										
011E 0005 ADD [DI],AL										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 FF FF										AD DE 1B 05 C5 06 00 00										
DS:0010 18 01 10 01 18 01 92 01										01 01 01 00 02 FF FF FF										
DS:0020 FF FF FF FF FF FF FF FF										FF FF FF FF EB 19 E6 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.n i . + ...										
									ff.										
										δ.μ.										
										ó.....J.										
																			

now the value is added in Ax and become 000A

- On next step more value is added in Ax.

- At next the interpret code mov

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

AX 000F	SI 0000	CS 19F5	IP 0110	Stack +0 0000	Flags 7204
BX 0000	DI 0000	DS 19F5		+2 20CD	
CX 0000	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 > █				000F	
010D A11D01	MOV	AX, [011D]			
0110 01061F01	ADD	[011F], AX			
0114 B8004C	MOV	AX, 4C00			
0117 CD21	INT	21			
0119 05000A	ADD	AX, 0A00			
011C 000F	ADD	[BX], CL			
011E 000F	ADD	[BX], CL			
0120 0000	ADD	[BX+SI], AL			
0122 008B46F6	ADD	[F646+BP+DI], CL			

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	FF	FF	AD	DE	1B	05	C5	06	00	00	= f.n i ..+...
DS:0010	18	01	10	01	18	01	92	01	01	01	01	00	02	FF	FF	FFff.
DS:0020	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	EB	19	E6	11	δ.p.
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

to ax and code will

terminate.

5) Understanding the c02-05.asm:

- This is the code inside the file of c02-05.asm

```

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

    mov ax, [num1]

    mov bx, [num1+1]
    add ax, bx

    mov bx, [num1+2]
    add ax, bx

    mov [num1+3], ax

    mov ax, 0x4C00
    int 0x21

num1: db 5, 10, 15, 0

; something's wrong with this code.
; let's figure out what that is!

```

- We can add values in this code one at a time. First, save the value in ax, then bx, then add both, and so on. Let's now examine it in the debugger.
- However, there is a problem since, as we know, db stores values in 1 bit, whereas ax stores values in 16 bits. As a result, you can see that there is an error when storing values in Ax, as shown in the figure.

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

AX 0A05	SI 0000	CS 19F5	IP 0103	Stack +0 0000	Flags 7200
BX 0000	DI 0000	DS 19F5		+2 20CD	
CX 001B	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 > **take two bits after opcode**

0100 A11701	MOV	AX, [0117]	DS:0000	CD 20 FF 9F 00 EA F0 FE
0103 8B1E1801	MOV	BX, [0118]	DS:0008	AD DE 1B 05 C5 06 00 00
0107 01D8	ADD	AX, BX	DS:0010	18 01 10 01 18 01 92 01
0109 8B1E1901	MOV	BX, [0119]	DS:0018	01 01 01 00 02 FF FF FF
010D 01D8	ADD	AX, BX	DS:0020	FF FF FF FF FF FF FF FF
010F A31A01	MOV	[011A], AX	DS:0028	FF FF FF FF EB 19 C0 11
0112 B8004C	MOV	AX, 4C00	DS:0030	A2 01 14 00 18 00 F5 19
0115 CD21	INT	21	DS:0038	FF FF FF FF 00 00 00 00
0117 050A0F	ADD	AX, 0F0A	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	FFft.
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	6.....J.
DS:0040	05	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

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

C02-02.COM C02-02.LST C02-03.COM C02-03.LST C02-04.COM

- It can store two different values instead of storing the just value 10. This is the issue in that code.

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

AX 0A05 SI 0000 CS 19F5 IP 0103 Stack +0 0000 Flags 7200
 BX 0000 DI 0000 DS 19F5 +2 20CD
 CX 001B 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 > take two bits after opcode

Address	Disassembly	Comment
0100	A11701	MOV AX, [0117]
0103	8B1E1801	MOV BX, [0118]
0107	01D8	ADD AX, BX
0109	8B1E1901	MOV BX, [0119]
010D	01D8	ADD AX, BX
010F	A31A01	MOV [011A], AX
0112	B8004C	MOV AX, 4C00
0115	CD21	INT 21
0117	050A0F	ADD AX, 0F0A

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 FFf.
 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 6.....J.
 DS:0040 05 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

C02-02.COM C02-02.LST C02-03.COM C02-03.LST C02-04.COM

6) Understanding the c02-06.asm:

- So now we see the solution of that previous one. Lets see the code of c02-06
- Here is the code of this file

```

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

    mov  ah, [num1]          ; Intel Software Developer

    mov  bh, [num1+1]
    add  ah, bh

    mov  bh, [num1+2]
    add  ah, bh

    mov  [num1+3], ah

    mov  ax, 0x4c00
    int  0x21

num1: db  5, 10, 15, 0

```

- Therefore, we can put values in the al or ah register rather than the ax register in order to avoid making this mistake. This is because the al and ah registers are 8-bit registers. Likewise for bl in place of bx. And this will address the issues we had with the earlier code.
- Let's test this in Dosbox to see whether it solves our issue.

Proceed to look for BL.

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

AX 0500	SI 0000	CS 19F5	IP 0108	Stack +0 0000	Flags 7200
BX 0A00	DI 0000	DS 19F5		+2 20CD	
CX 001D	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 >

0104 8A3E1A01	MOV	BH, [011A]	DS:0000	CD 20 FF 9F 00 EA F0 FE
0108 00FC	ADD	AH, BH	DS:0008	AD DE 1B 05 C5 06 00 00
010A 8A3E1B01	MOV	BH, [011B]	DS:0010	18 01 10 01 18 01 92 01
010E 00FC	ADD	AH, BH	DS:0018	01 01 01 00 02 FF FF FF
0110 88261C01	MOV	[011C], AH	DS:0020	FF FF FF FF FF FF FF
0114 B8004C	MOV	AX, 4C00	DS:0028	FF FF FF FF EB 19 C0 11
0117 CD21	INT	21	DS:0030	A2 01 14 00 18 00 F5 19
0119 050A0F	ADD	AX, 0F0A	DS:0038	FF FF FF FF 00 00 00 00
011C 00E6	ADD	DH, AH	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 FFff.
DS:0020	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

num1: db 5, 10, 15, 0

Value 10 appears to be perfectly stored in the Bx register. The remainder of the code operates flawlessly as well because all addition and subtraction will function properly if the value is entered correctly.AL REGISTER AND WE USE THE SAME CODE.

7) Being aware of the c02-07.asm:

This is the C02-07.asm code. This code uses add values in bx to jump, and we know that the [] pick value comes from the location enclosed in curly brackets. We now write bx, move it into axe, and leap by simply adding bx's address.

```

[org 0x0100]

    mov     bx, num1
    add     ax, [bx]
    add     bx, 2
    add     ax, [bx]
    add     bx, 2
    add     ax, [bx]
    add     bx, 2

    mov     [result], ax
    mov     ax, 0x4c00
    int     0x21

|
num1: dw    5, 10, 15
result: dw  0

```

- Bx has the address of num1 value 5.
- we can increase the value according to the type and get the next value store in num1.

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD															-	×								
AX 0000	SI 0000	CS 19F5	IP 0103	Stack +0	0000	Flags 7200																		
BX 011D	DI 0000	DS 19F5		+2	20CD																			
CX 0000	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 >																								
0005																								
0100 BB1D01	MOV	BX,011D													1	0	1	2	3	4	5	6	7	
0103 0307	ADD	AX,[BX]													DS:0000	CD	20	FF	9F	00	EA	FF	FF	
0105 81C30200	ADD	BX,0002													DS:0008	AD	DE	1B	05	C5	06	00	00	
0109 0307	ADD	AX,[BX]													DS:0010	18	01	10	01	18	01	92	01	
010B 81C30200	ADD	BX,0002													DS:0018	01	01	01	00	02	FF	FF	FF	
010F 0307	ADD	AX,[BX]													DS:0020	FF	FF	FF	FF	FF	FF	FF	FF	
0111 81C30200	ADD	BX,0002													DS:0028	FF	FF	FF	FF	EB	19	E6	11	
0115 A32301	MOV	[0123],AX													DS:0030	A2	01	14	00	18	00	F5	19	
0118 B8004C	MOV	AX,4C00													DS:0038	FF	FF	FF	FF	00	00	00	00	
															DS:0040	05	00	00	00	00	00	00	00	00
															DS:0048	00	00	00	00	00	00	00	00	00
2																								
DS:0100	BB	1D	01	03	07	81	C3	02	00	03	07	81	C3	02	00	03	η...ü†... ..ü†...							
DS:0110	07	81	C3	02	00	A3	23	01	B8	00	4C	CD	21	05	00	0A	.ü†.ú# 7.L=?...							
DS:0120	00	0F	00	1E	00	F6	D1	E0	D1	E0	C5	5E	D8	01	C3	8B÷τα τα†^†.†i							
DS:0130	07	8B	57	02	85	D2	75	04	85	C0	74	1C	C7	46	DC	00	.iW.àπu. àLt. F■.							
DS:0140	00	8E	5E	FC	83	7D	0E	00	74	09	8B	46	F2	48	3B	46	.Ä^nâ).. t.iF2H:F							

- Address of num1 save in bx and we see value of bx changed.

- Now after apply braces its take the value on this address and store it into ax.
- Now add that bx value into ax

```

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD
AX 000F SI 0000 CS 19F5 IP 010B Stack +0 0000 Flags 7204
BX 011F DI 0000 DS 19F5 +2 20CD
CX 0000 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 >

0109 0307 ADD AX,[BX]
010B 81C30200 ADD BX,0002
010F 0307 ADD AX,[BX]
0111 81C30200 ADD BX,0002
0115 A32301 MOV [0123],AX
0118 B8004C MOV AX,4C00
011B CD21 INT 21
011D 05000A ADD AX,0A00
0120 000F ADD [BX],CL

1 0 1 2 3 4 5 6 7
DS:0000 CD 20 FF 9F 00 EA FF FF
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 E6 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:0100 BB 1D 01 03 07 81 C3 02 00 03 07 81 C3 02 00 03
DS:0110 07 81 C3 02 00 A3 23 01 B8 00 4C CD 21 05 00 0A
DS:0120 00 0F 00 1E 00 F6 D1 E0 D1 E0 C5 5E D8 01 C3 8B
DS:0130 07 8B 57 02 85 D2 75 04 85 C0 74 1C C7 46 DC 00
DS:0140 00 8E 5E FC 83 7D 0E 00 74 09 8B 46 F2 48 3B 46

```

- Result will now see at Ax.
- Now next address in bx and its value is store in ax.

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD																	
AX 001E	SI 0000	CS 19F5	IP 0111	Stack +0	0000	Flags 7214											
BX 0121	DI 0000	DS 19F5		+2	20CD												
CX 0000	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 1 1 0											
CMD >							1	0	1	2	3	4	5	6	7		
010F 0307		ADD	AX,[BX]	DS:0000	CD 20 FF 9F 00 EA FF FF												
0111 81C30200		ADD	BX,0002	DS:0008	AD DE 1B 05 C5 06 00 00												
0115 A32301		MOV	[0123],AX	DS:0010	18 01 10 01 18 01 92 01												
0118 B8004C		MOV	AX,4C00	DS:0018	01 01 01 00 02 FF FF FF												
011B CD21		INT	21	DS:0020	FF FF FF FF FF FF FF FF												
011D 05000A		ADD	AX,0A00	DS:0028	FF FF FF FF EB 19 E6 11												
0120 000F		ADD	[BX],CL	DS:0030	A2 01 14 00 18 00 F5 19												
0122 001E00F6		ADD	[F600],BL	DS:0038	FF FF FF FF 00 00 00 00												
0126 D1E0		SHL	AX,1	DS:0040	05 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:0100	BB	1D	01	03	07	81	C3	02	00	03	07	81	C3	02	00	03	η...ü ...ü ...
DS:0110	07	81	C3	02	00	A3	23	01	B8	00	4C	CD	21	05	00	0A	ü ...ü#...7.L=?...
DS:0120	00	0F	00	1E	00	F6	D1	E0	D1	E0	C5	5E	D8	01	C3	8B÷ααα^††.†i
DS:0130	07	8B	57	02	85	D2	75	04	85	C0	74	1C	C7	46	DC	00	.iW.àαu.à^t.αF■.
DS:0140	00	8E	5E	FC	83	7D	0E	00	74	09	8B	46	F2	48	3B	46	.â^uâ}...t.iF2H:F

- Result after store into Ax.
- Next store the ax address into another variable.

```

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: AFD
AX 001E SI 0000 CS 19F5 IP 0115 Stack +0 0000 Flags 7200
BX 0123 DI 0000 DS 19F5 +2 20CD
CX 0000 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 >

0111 81C30200 ADD BX,0002
0115 A32301 MOV [0123],AX
0118 B8004C MOV AX,4C00
011B CD21 INT 21
011D 05000A ADD AX,0A00
0120 000F ADD [BX],CL
0122 001E00F6 ADD [F600],BL
0126 D1E0 SHL AX,1
0128 D1E0 SHL AX,1

1 0 1 2 3 4 5 6 7
DS:0000 CD 20 FF 9F 00 EA FF FF
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 E6 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:0100 BB 1D 01 03 07 81 C3 02 00 03 07 81 C3 02 00 03
DS:0110 07 81 C3 02 00 A3 23 01 B8 00 4C CD 21 05 00 0A
DS:0120 00 0F 00 1E 00 F6 D1 E0 D1 E0 C5 5E D8 01 C3 8B
DS:0130 07 8B 57 02 85 D2 75 04 85 C0 74 1C C7 46 DC 00
DS:0140 00 8E 5E FC 83 7D 0E 00 74 09 8B 46 F2 48 3B 46

```

- Then at last the interept command and the program will terminate.