Name: Tazmeen Afroz Roll No: 22P-9252 Section: BAI-4A Coal Lab Task 9

Problem: Multiplying Larger Numbers

When multiplying numbers, we face a limitation when dealing with larger numbers that exceed the register size (16 bits or 32 bits). Standard shifting operations (SHL, SHR) can only manipulate bits within the register size, causing significant bits to be dropped or lost when shifting larger numbers. This results in incorrect products. For example: When multiplying two 16-bit numbers, the result can be up to 32 bits long. However, we only have 16-bit registers available. This means we need to find a way to store and process the partial products without overflowing the registers.

Solution: Extended Shifting

To overcome the memory issue, we use extended shifting.

The Algorithm

The extended shifting algorithm involves two key instructions: SHL (Shift Left) and RCL (Rotate Carry Left). This technique is used to shift a 32-bit number left by 16 bits, ensuring no valuable bit is lost during the operation.

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

In this example, num1 is a 32-bit number stored in memory. The SHL instruction shifts the lower 16 bits of num1 to the left, and the most significant bit is dropped into the carry. The RCL instruction then pushes this bit into the least significant bit of the next word, effectively joining the two 16-bit words.

For **shifting right,** the process is reversed. The SHR (Shift Right) and RCR instructions are used to ensure that no valuable bit is lost.

num1: dd 40000 shr word [num1+2], 1 rcr word [num1], 1 word [num1], 1

- ADC (Add with Carry):

- Adds two numbers and the carry flag.

CODE

```
[org 0x0100]
imp start
multiplicand: dd 1300 ; 16bit multiplicand 32bit space
multiplier: dw 500 ; 16bit multiplier
result: dd 0 ; 32bit result
start: mov cl, 16; initialize bit count to 16
mov dx, [multiplier]; load multiplier in dx
checkbit: shr dx, 1; move right most bit in carry
jnc skip; skip addition if bit is zero
 mov ax, [multiplicand]
 add [result], ax; add less significant word
 mov ax, [multiplicand+2]
 adc [result+2], ax; add more significant word
skip: shl word [multiplicand], 1
 rcl word [multiplicand+2], 1; shift multiplicand left
 dec cl; decrement bit count
 jnz checkbit; repeat if bits left
 mov ax, 0x4c00
```

Code Logic:

- 1. Initialize bit count to 16 and load multiplier in DX.
 - Set CL to 16 to keep track of the number of bits in the multiplier.
 - Load the multiplier into DX.
- 2. Check the rightmost bit of the multiplier (DX). If 0, skip addition.
 - Shift DX right by one bit, moving the rightmost bit to the carry.
 - If the carry is not set (meaning the rightmost bit of the multiplier was 0), proceed to step 4.
- 3. If the rightmost bit of the multiplier was 1, proceed with addition:
 - Add the less significant word of the multiplicand to the result.
 - Add the more significant word of the multiplicand to the result with carry.
- 4. Shift the multiplicand left by one bit.

- Shift the lower 16 bits of multiplicand left by one bit, moving the leftmost bit to the carry and filling the rightmost bit with 0.
- 5. Rotate the multiplier right by one bit with carry.
 - Rotate the carry left into the least significant bit of the upper 16 bits of multiplicand.
- 6. Decrement the bit count and repeat steps until all bits are processed.
 - · Decrement CL by one.
 - If CL is not zero, repeat steps 2-5.
- 7. **Finalization**: After all iterations are complete, the result contains the 32-bit product of the two 16-bit numbers.

Binary Representation:

Multiplicand (1300): 0000010110100100 Multiplier (500): 0000000110011000

Step-Wise Code Explanation: Initialization:

[org 0x0100]; Origin at memory address 100h jmp start; Jump to the start of the program

Data Definition:

multiplicand: dd 1300; Define a 32-bit data element (double word)
The dd 1300 directive reserves 4 bytes of memory. The value 1300 in hexadecimal is **0x0514.**Since it's a 16-bit value, it's stored in the lower 16 bits of the 32-bit space, with the upper 16

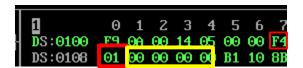
bits being zero.



multiplier: dw 500; Define a 16-bit data element (word)

Similar to line 3, dw 500 reserves 2 bytes. The value 500 in hexadecimal is 0x01F4.

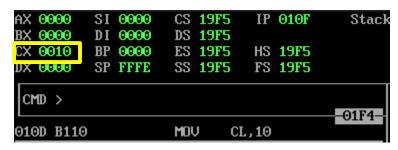
result: dd 0; Define a 32-bit data element (double word) to store the result



Program Start

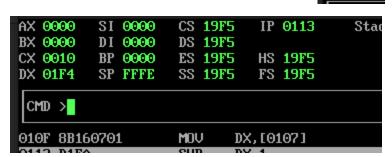
start:

mov cl, 16; Initialize the bit count to 16 (0010 in hex)



DX 01F4

mov dx, [multiplier]; Load the multiplier into the dx register



Iteration Loop First Iteration:

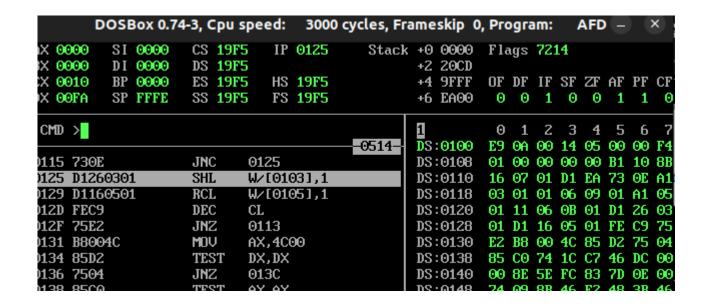
checkbit:

shr dx, 1; Shift the multiplier right by one bit and move the rightmost bit to the carry $(111110100 \rightarrow 011111010)$ (hex will be 0xFA)(Decimal $\rightarrow 250$)



jnc skip; skip the addition

As carry flag is not set we are skipping the addition.



skip: shl word [multiplicand], 1

This instruction shifts the bits in the lower 16 bits of multiplicand to the left by one position. The leftmost bit is moved to the carry flag, and the rightmost bit is filled with a 0.

• Before: 0000010110100100

• After: 0000101000101000 (binary representation after shifting left by one) in hex A28



rcl word [multiplicand+2], 1;

This instruction rotates the carry left into the least significant bit of the upper 16 bits of multiplicand. Since the carry flag was not set by the previous shift operation, it remain same

- Before: 000000000000 (upper 16 bits of multiplicand before rotation)
- After: 0000000000 (upper 16 bits of multiplicand after rotation)

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

Iteration 2:

DOSBox 0.74	-3, Cpu s	peed:	3000	cycles, Fr	ame	skip 0	, Pro	одга	ım:	-	AFD	E) (×
AX 0000 SI 0000	CS 19F		012D	Stack			Flā	ags	721	L4				
BX 0000 DI 0000 CX 000F BP 0000	DS 19F9 ES 19F9		19F5		_	20CD 9FFF	OF	DF	ΙF	SF	ZF	ΑF	PF	CF
DX 007D SP FFFE	SS 19F		19F5			EA00	0	0	1	0	0	1	1	Θ
CMD >					1		0	1	2	3	4	5	6	7
						0100	E9	ΘA	00	50	14	00	00	F4
0129 D1160501	RCL	W/[010	951,1		DS:	0108	01	$\Theta\Theta$	$\Theta\Theta$	90	90	B1	10	8B
012D FEC9	DEC	CL			DS:	0110	16	07	01	D1	ΕA	73	ΘE	A1
012F 75E2	JNZ	0113			DS:	0118	03	01	01	96	09	01	A1	05 ⁰
0131 B8004C	MOV	AX,4C	90		DS:	0120	01	11	06	ΘB	01	D1	26	03_{0}
0134 85D2	TEST	DX,DX			DS:	0128	01	D1	16	05	01	FΕ	C9	75
0136 750 4	JNZ	013C			DS:	0130	EZ	B8	00	4 C	85	DZ	75	04
0138 8500	TEST	AX,AX			DS:	0138	85	CΘ	74	10	C7	46	DC	ΘΘ
013A 741C	JZ	0158			DS:	0140	99	8E	5E	FC	83	7D	ΘΕ	ΘΘ
013C C746DC0000	MOV	[BP-24	41,0000	9	DS:	0148	74	09	8B	46	FZ	48	3B	46
								_						

Iteration 3: multiplicand : 000000001111101 After shift right carry flag is set.

	DOSE	3ox 0.7	4-3, C	pu s	peed:	3000	cycles,	Fram	ie	skip 0,	Pro	ogra	m:	,	AFD) (× S
AX 0000		0000		19F5		0115	Sta			0000	Fla	ags	72:	11				
BX 0000 CX 000E		0000 0000		19F5 19F5		19F5		+		20CD 9FFF	OF	DΕ	IF	SE.	ΖF	ΔF	PF	CE
0X 003E		FFFE		19F5		19F5				EA00	0	0	1	0	0	1	0	1
CMD >								1			0	1	2	3	4	5	6	7
										0100	E9	0A	00	50	14	00	00	F4
9113 D11	EA		SHF	}	DX,1			D:	s :	0108	01	00	00	00	00	B1	10	8B
9115 730	9E		JNC)	0125			D	S :	0110	16	07	01	D1	ΕA	73	ΘE	A1 '
9117 A10	9301		MOL	J	AX,[0:	1031		D:	S :	0118	03	01	01	96	09	01	A1	0 5_
911A 010	960901	1	ADI)	[0109]	XA,[D	s :	0120	01	11	06	ΘB	01	D1	26	03
911E A10	0501		MOL	J	AX,[0:	1051		D	s :	0128	01	D1	16	05	01	FЕ	C9	75 (
9121 110	960B01	1	ADC)	[010B]],AX		D	s :	0130	EZ	B8	00	4 C	85	D2	75	04
9125 D12	260301	1	SHI	4	W/[010	931,1		D	s:	0138	85	CO	74	1 C	C7	46	DC	00
9129 D11	160501	1	RCI	4	W/[010	951,1		D	s:	0140	00	8E	5E	FC	83	7D	ΘE	00
912D FE0	C9		DEC		CL			D	s :	0148	74	09	8B	46	FZ	48	3B	46

Addition:

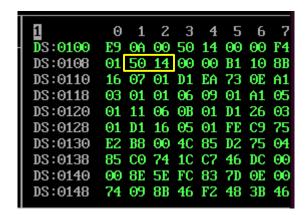
mov ax, [multiplicand] add [result], ax; add less significant word mov ax, [multiplicand+2] adc [result+2], ax; add more significant word

mov the multiplicand to ax

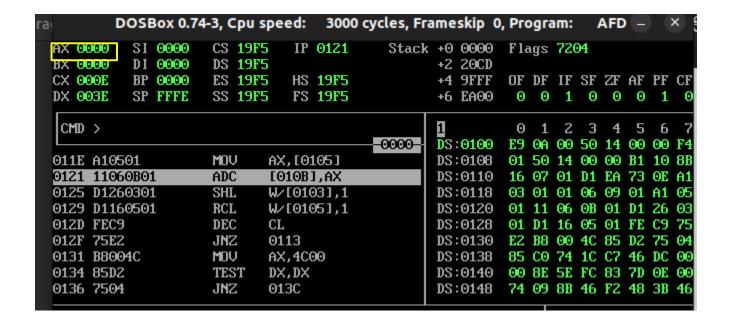


multiplicand 0001010001010000

add [result], ax;



mov ax, [multiplicand+2]



adc [result+2], ax; add more significant word.

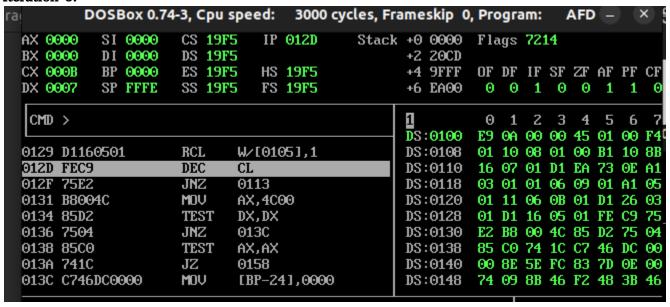
ra DOSBox 0.74	-3, Cpu s	peed:	3000	cycles, Fra	ames	kip 0,	Рισ	одга	m:	-	AFD	\subseteq		×) €
AX 0000 SI 0000	CS 19F5		0125	Stack			Fla	ıgs	724	14				
BX 0000 DI 0000	DS 19F5				+2	20CD								
CX 000E BP 0000	ES 19F5	HS	19F5		+4	9FFF I	\mathbf{OF}	DF	\mathbf{IF}	SF	ΖF	ΑF	\mathbf{PF}	\mathbf{CF}_{\cdot}
DX 003E SP FFFE	SS 19F5	FS	19F5		+6	EAOO	0	Θ	1	Θ	1	Θ	1	Θ.
CMD >				-	1		0	1	2	3	4	5	6	7 ⁸
				-1450-		0100	E9	ΘA	00	50	14	90	00	F4 ³
0121 11060B01	ADC	[010B]	AX.		DS:	0108	01	50	14	00	00	B1	10	8B
0125 D1260301	SHL	W/[010	931,1		DS:	0110	16	07	01	D1	EΑ	73	ΘΕ	A1.
0129 D1160501	RCL	W/[010	951,1		DS:	0118	03	01	01	96	09	01	A1	05 3
012D FEC9	DEC	CL			DS:	0120	01	11	06	ΘB	01	D1	26	03^{L}_{1}
012F 75E2	JNZ	0113			DS:	0128	01	D1	16	05	01	FE	C9	75 3
0131 B8004C	MOV	AX,4CC	90		DS:	0130	EZ	B8	00	4 C	85	DZ	75	04 ¹
0134 85D2	TEST	DX,DX			DS:	0138	85	CO	74	1 C	C7	46	DC	$00^{\rm L}$
0136 7504	JNZ	013C			DS:	0140	00	8E	5E	FC	83	7D	ΘE	00^{l}
0138 8500	TEST	AX,AX			DS:	0148	74	09	8B	46	FZ	48	3B	46 ²

ra; DOSBox	c 0.74-3, Cpu speed	: 3000 cycles, Fr	ameskip 0, 1	Program:	AF	9 (-)	×
AX 0000 SI 00		P 012D Stack		flags 72	14		
BX 0000 DI 00			+2 20CD				
CX 000E BP 00		S 19F5			SF ZF	AF P	F CF
DX 003E SP FF	FE SS 19F5 F	S 19F5	+6 EA00	0 0 1	0 0	1	1 O _L
CMD >			1	0 1 2	3 4	5	6 7
				E9 0A 00	A0 28	99 9	Θ F4 ₁
0129 D1160501	RCL W∕[0	1051,1	DS:0108 (91 50 14	00 00	B1 1	⊙ 8Bi
012D FEC9	DEC CL		DS:0110 1	16 07 01	D1 EA	73 0	E A15
012F 75E2	JNZ 0113		DS:0118 0	93 01 01	06 09	01 A	1 05
0131 B8004C	MOV AX,4	C00	DS:0120 0	91 11 06	OB 01	D1 2	6 03
0134 85D2	TEST DX,D	X	DS:0128 0	91 D1 16	05 01	FE C	9 75,
0136 7504	JNZ 013C		DS:0130 I	EZ B8 00	4C 85	D2 7	5 04
0138 85C0	TEST AX,A	X	DS:0138 8	35 CO 74	1C C7	46 D	C 00
013A 741C	JZ 0158		DS:0140 (90 8E 5E	FC 83	7D 0	E 00
013C C746DC0000	MOV [BP-7	241,0000	DS:0148	74 09 8B	46 F2	48 3	B 46

Iteration 4:

raj [OOSBox 0.74	-3, Cpu s	peed:	3000	cycles, Fra	ame	skip 0,	Pro	одга	m:	-	AFD	<u> </u>		×) {
AX 0000 BX 0000	SI 0000 DI 0000	CS 19F9		012D	Stack		0000 20CD	Fla	ags	729	90				
CX 000C DX 000F	BP 0000 SP FFFE	ES 19F	HS	19F5		+4	9FFF EAOO	OF O	DF O	IF	SF	ZF 0	ΑF		CF
DX 0001	21 111	SS 19F	61 (19F5		+0	EHOO							0	
CMD >						1	.0400	0	1	Z			5		_
0129 D116	50501	RCL	W/[010	951.1			:0100 :0108		90						_
O12D FECS		DEC	CL						07						_
012F 75E2	_	JNZ	0113	~			:0118		01						
0131 B800 0134 85D2		MOV TEST	AX,4CO				:0120 :0128		11 D1						_
0136 7504	1	JNZ	013C			DS	:0130	EZ	B8	00	4 C	85	D2	75	04
0138 8500	_	TEST	AX,AX				:0138		CO						_
013A 7410 013C C746		JZ MOV	0158 [BP-24	11,000	э		:0140 :0148		8E 09						00 46

Iteration 5:



Iteration 6:

ittiation o.															
га	DOSBox 0.74	-3, Cpu s	peed:	3000	cycles, Fra	ame	skip 0,	Pro	одга	ım:		AFD			×) {
AX 0001	SI 0000	CS 19F	5 IP	012D	Stack	+0	0000	Fla	ags	729	94				
BX 0000	DI 0000	DS 19F	5			+2	20CD								
CX 000A	BP 0000	ES 19F	5 HS	19F5		+4	9FFF	\mathbf{OF}	DF	\mathbf{IF}	SF	ΖF	ΑF	PF	\mathbf{CF}
DX 0003	SP FFFE	SS 19F	5 FS	19F5		+6	EA00	0	0	1	1	0	1	1	0
CMD >						1		0	1	2	3	4	5	6	7
							0100	E9	ΘA		00	8A	02	00	F4.
0129 D1	160501	RCL	W/[010	951,1			:0108		10						
012D FE	C9	DEC	CL			DS	:0110	16	07	01	D1	ΕA	73	ΘΕ	A1
012F 75	EZ	JNZ	0113			DS	:0118	03	01	01	06	09	01	A1	05
0131 B8	004C	MOV	AX,4CC	90		DS	:0120	01	11	06	ΘB	01	D1	26	03
0134 85	D2	TEST	DX,DX			DS	:0128	01	D1	16	05	01	FΕ	C9	75
0136 75	04	JNZ	013C			DS	:0130	EZ	B8	99	4 C	85	DZ	75	04
0138 85	CO	TEST	AX,AX			DS	:0138	85	CO	74	1 C	C7	46	DC	00
013A 74	1C	JZ	0158			DS	:0140	00	8E	5E	FC	83	7D	ΘE	00
013C C7	46DC0000	MOV	[BP-24	11,000	0	DS	:0148	74	09	8B	46	FZ	48	3B	46

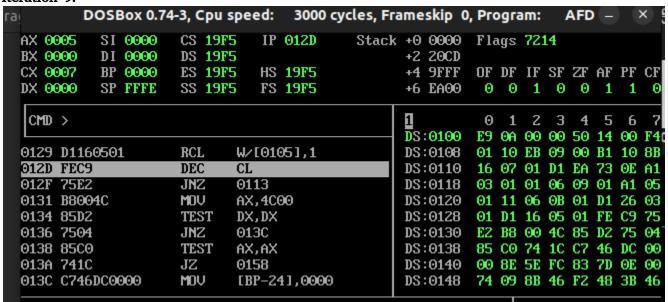
Iteration 7:

raı DOSBox 0.74	-3, Cpu speed	l: 3000 cy	cles, Fram	eskip 0,	Progra	im:	AFD -) (× {
AX 0002 SI 0000		P 012D	Stack +6		Flags	7214			
BX 0000 DI 0000 CX 0009 BP 0000	DS 19F5 ES 19F5 H	IS 19F5	_	20CD 19FFF	OF DF	IF SF	ZF AI	F PF	CF
DX 0001 SP FFFE		S 19F5		EA00	0 0	1 0	0	1 1	ο̈
CMD >			1		0 1	2 3	4 !	5 6	
CID 7				:0100	E9 0A				
0129 D1160501	RCL W∕[€	01051,1			01 10				
012D FEC9	DEC CL		DS	:0110	16 07	01 D1	EA 7	3 OE	A1
012F 75E2	JNZ 0113	}	DS	3:0118	03 01	01 06	09 0	1 A1	05
0131 B8004C	MOV AX,4	łC00	DS	:0120	01 11	06 OB	01 D:	1 26	03
0134 85D2	TEST DX,D	X	DS	3:0128	01 D1	16 05	01 F	E C9	75
0136 750 4	JNZ 0130		DS	:0130	E2 B8	00 4C	85 D	2 75	04
0138 85C0	TEST AX,A	ìΧ	DS	:0138	85 CO	74 1C	C7 4	5 DC	00
013A 741C	JZ 0158	3	DS	:0140	00 SE	5E FC	83 7	OE	00
013C C746DC0000	MOV [BP-	241,0000	DS	3:0148	74 09	8B 46	F2 4	3 B	46

Iteration 8:

AX 0005 SI 0000 CS 19F5 IP 012D Stack +0 0000 Flags 7214 BX 0000 DI 0000 DS 19F5 CX 0008 BP 0000 ES 19F5 HS 19F5 DX 0000 SP FFFE SS 19F5 FS 19F5 CMD > CMD > 1 0 1 2 3 4 5 6 7 DS:0100 E9 0A 00 00 28 0A 00 F4 DS:0100 E9 0A 00 00 28 0A 00 F4 DS:0100 E9 0A 00 00 28 0A 00 F4 DS:0110 16 07 01 D1 EA 73 0E A1 DS:0131 B8004C MDU AX,4C00 0134 B5D2 TEST DX,DX DS:0130 E2 B8 00 4C 85 D2 75 0136 7504 DS:0138 B5C0 TEST AX,AX DS:0138 B5 C0 74 1C C7 46 DC 00 0130 C746DC00000 MDU IBP-241,0000 DS:0148 74 09 8B 46 F2 48 3B 46	га	DOSBox 0.74	-3, Cpu s	peed:	3000	cycles, Fra	ame	skip 0,	Pro	одга	m:	-	AFD	Ē		×) {
CX 0008 BP 0000 ES 19F5 HS 19F5					012D	Stack			Flā	ags	721	14				
CMD > 1					19F5				OF	DF	IF	SF	ZF	ΑF	PF	CF
DS:0100 E9 0A 00 00 28 0A 00 F40	DX 0000	SP FFFE	SS 19F	FS	19F5		+6	EA00	0	0	1	0	0	1	1	Θ
0129 D1160501 RCL W/[0105],1 DS:0108 01 10 EB 09 00 B1 10 8B 012D FEC9 DEC CL DS:0110 16 07 01 D1 EA 73 0E A1 012F 75E2 JNZ 0113 DS:0118 03 01 01 06 09 01 A1 05 0131 B8004C MOV AX,4C00 DS:0120 01 11 06 0B 01 D1 26 03 0134 85D2 TEST DX,DX DS:0128 01 D1 16 05 01 FE C9 75 0136 7504 JNZ 013C DS:0130 E2 B8 00 4C 85 D2 75 04- 0138 85C0 TEST AX,AX DS:0138 85 C0 74 1C C7 46 DC 00 013A 741C JZ 0158 DS:0140 00 8E 5E FC 83 7D 0E 00	CMD >								_	_				_	_	7
912D FEC9 DEC CL DS:0110 16 07 01 D1 EA 73 0E A1 012F 75E2 JNZ 0113 DS:0118 03 01 01 06 09 01 A1 05 0131 B8004C MOU AX,4C00 DS:0120 01 11 06 0B 01 D1 26 03 0134 85D2 TEST DX,DX DS:0128 01 D1 16 05 01 FE C9 75 0136 7504 JNZ 013C DS:0130 E2 B8 00 4C 85 D2 75 04- 0138 85C0 TEST AX,AX DS:0138 85 C0 74 1C C7 46 DC 00 013A 741C JZ 0158 DS:0140 00 8E 5E FC 83 7D 0E 00							DS	0100								
012F 75E2 JNZ 0113 DS:0118 03 01 01 06 09 01 A1 05 0131 B8004C MOU AX,4C00 DS:0120 01 11 06 0B 01 D1 26 03 0134 85D2 TEST DX,DX DS:0128 01 D1 16 05 01 FE C9 75 0136 7504 JNZ 013C DS:0130 E2 B8 00 4C 85 D2 75 04 0138 85C0 TEST AX,AX DS:0138 85 C0 74 1C C7 46 DC 00 013A 741C JZ 0158 DS:0140 00 8E 5E FC 83 7D 0E 00	0129 D11	160501	RCL	W/[010	95],1		DS	:0108	01	10	$\mathbf{E}\mathbf{B}$	09	00	B1	10	8B
0131 B8004C MOV AX,4C00 DS:0120 01 11 06 0B 01 D1 26 03 0134 85D2 TEST DX,DX DS:0128 01 D1 16 05 01 FE C9 75 0136 7504 JNZ 013C DS:0130 E2 B8 00 4C 85 D2 75 04- 0138 85C0 TEST AX,AX DS:0138 85 C0 74 1C C7 46 DC 00 013A 741C JZ 0158 DS:0140 00 8E 5E FC 83 7D 0E 00	012D FEO	29	DEC	CL			DS	:0110	16	07	01	D1	ΕA	73	ΘE	A1
0134 85D2 TEST DX,DX DS:0128 01 D1 16 05 01 FE C9 75 0136 7504 JNZ 013C DS:0130 E2 B8 00 4C 85 D2 75 04- 0138 85C0 TEST AX,AX DS:0138 85 C0 74 1C C7 46 DC 00 013A 741C JZ 0158 DS:0140 00 8E 5E FC 83 7D 0E 00	012F 75E	EZ	JNZ	0113			DS	:0118	03	01	01	06	09	01	A1	05
0136 7504 JNZ 013C DS:0130 E2 B8 00 4C 85 D2 75 04- 0138 85C0 TEST AX,AX DS:0138 85 C0 74 1C C7 46 DC 00 013A 741C JZ 0158 DS:0140 00 8E 5E FC 83 7D 0E 00	0131 B86	904C	MOV	AX,4CC	90		DS	:0120	01	11	06	ΘB	01	D1	26	03
0138 85C0 TEST AX,AX DS:0138 85 C0 74 1C C7 46 DC 00 013A 741C JZ 0158 DS:0140 00 8E 5E FC 83 7D 0E 00	0134 850	02	TEST	DX,DX			DS	:0128	01	D1	16	05	01	\mathbf{FE}	C9	75
013A 741C JZ 0158 DS:0140 00 8E 5E FC 83 7D 0E 00	0136 750	94	JNZ	013C			DS	:0130	E2	B8	00	4 C	85	DZ	75	04
	0138 850	00	TEST	AX,AX			DS	:0138	85	CO	74	1 C	C7	46	DC	00
013C C746DC0000 MOV [BP-24],0000 DS:0148 74 09 8B 46 F2 48 3B 46	013A 741	LC	JZ	0158			DS	:0140	00	8E	5E	FC	83	7D	ΘE	00
	013C C74	16DC0000	MOV	[BP-24	11,000	9	DS	:0148	74	09	8B	46	FZ	48	3B	46

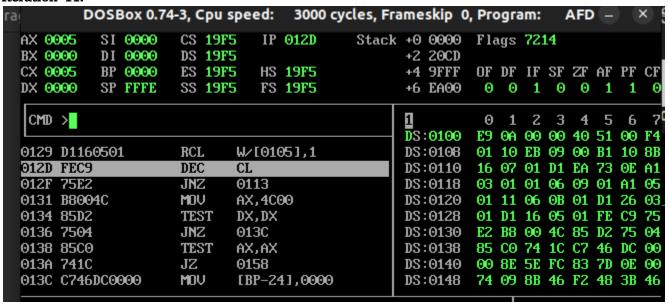
Iteration 9:



Iteration 10:

ra: DOSBox 0.74	-3, Cpu s	peed:	3000 cy	cles, Fra	ame	skip 0,	Pro	одга	m:	-	AFD	<u></u>		×) {
AX 0005 SI 0000 BX 0000 DI 0000	CS 19F5 DS 19F5		012D	Stack		0000 20CD	Fla	ags	729	94				
CX 0006 BP 0000 DX 0000 SP FFFE	ES 19F5 SS 19F5		19F5 19F5		_	9FFF EAOO	OF O	DF O	IF 1	SF 1	ZF 0	AF 1	PF 1	CF O
CMD >					100	:0100	0 E9	1	2		4 60		6	
0129 D1160501		W∕[010	51,1		DS	:0108	01	10	EB	09	00	B1	10	8B
012D FEC9 012F 75E2	JNZ	CL 0113				:0110 :0118	03	01	01	06	EA 09	01	A1	05
0131 B8004C 0134 85D2	MOV TEST	AX,4CO DX,DX	0			:0120 :0128					01 01			
0136 7504 0138 8500	JNZ TEST	013C AX,AX				:0130 :0138					85 C7			
013A 741C 013C C746DC0000		0158	1,0000		DS	:0140 :0148	00	8E	5E	FC	83 F2	7D	ΘE	00
								-						

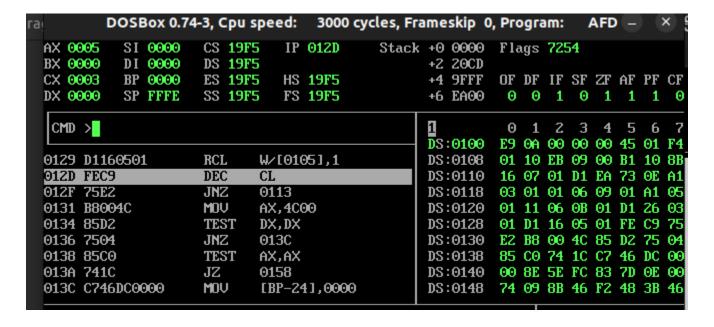
Iteration 11:



Iteration 12:

rai DO	SBox 0.74-3, C	pu speed:	3000 cyc	les, Fr	ameskip	0, Pro	ogran	n:	AFD		×	⟨ {
			012D	Stack	+0 0000		ags 7	294				
		19F5 19F5 HS	19F5		+2 20CI +4 9FFI		DF 1	F SF	ZF	ΔF	DF (CF
			19F5		+6 EAO			1 1	9	1	1	0
OMB >				$\neg \neg$	7				_	_		
CMD >					1 DS:0100	0 E9		23 0000			_6 ലൈ	F4.
0129 D11605	501 RCL	W/[010	951.1		DS:0100			ж В 09				
912D FEC9	DEC		,,, <u>,</u>		DS:0110			1 D1				
012F 75E2	JNZ	0113			DS:0118			1 06				
0131 B80040	MOV.	AX,400	90		DS:0120	01	11 6	6 OB	01	D1	26	03
0134 85D2	TES'	T DX,DX			DS:0128			6 05				
0136 7504	JNZ	013C			DS:0130	E2	B8 (00 40	85	DZ	75 (04
0138 85C0	TES'	T AX,AX			DS:0138	85	CO 7	4 10	C7	46	DC	00
013A 741C	JZ	0158			DS:0140	00	8E 5	E FC	83	7D	0E	00
013C C746DC	00000 MOV	[BP-24	11,0000		DS:0148	3 74	09 8	B 46	FZ	48	3B ·	46

Iteration 13:



Iteration 14:

DOSBox 0.74	4-3, Cpu speed:	3000 cycles, Fr	rameskip 0,	Progra	m: A	\FD -	× S
AX 0005 SI 0000 BX 0000 DI 0000	CS 19F5 IP DS 19F5	012D Stack	+0 0000 +2 20CD	Flags	7254		1
CX 0002 BP 0000		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 1	1 0
CMD >			1	0 1	2 3	4 5	
			DS:0100	E9 0A	00 00	00 BA	02 F4"
0129 D1160501	RCL W/[01	051,1	DS:0108	01 10	EB 09	00 B1	10 8B
012D FEC9	DEC CL		DS:0110	16 07	01 D1	EA 73	0E A1
012F 75E2	JNZ 0113		DS:0118	03 01	01 06	09 01	A1 05
0131 B8004C	MOV AX,40	00	DS:0120	01 11	06 OB	01 D1	26 03
0134 85D2	TEST DX,DX		DS:0128	01 D1	16 05	01 FE	C9 75—
0136 7504	JNZ 013C		DS:0130	E2 B8	00 4C	85 D2	75 04
0138 8500	TEST AX,AX		DS:0138	85 CO	74 10	C7 46	DC 00
013A 741C	JZ 0158		DS:0140	00 BE	5E FC	83 7D	0E 00
013C C746DC0000	MOV [BP-2	41,0000	DS:0148	74 09	8B 46	FZ 48	3B 46

Iteration 15:

