Practical Foundations of Debugging for x64 platforms

Part 2: Number representations and pointers

Designed, written and illustrated by Dmitry Vostokov

Review - MASM64 program

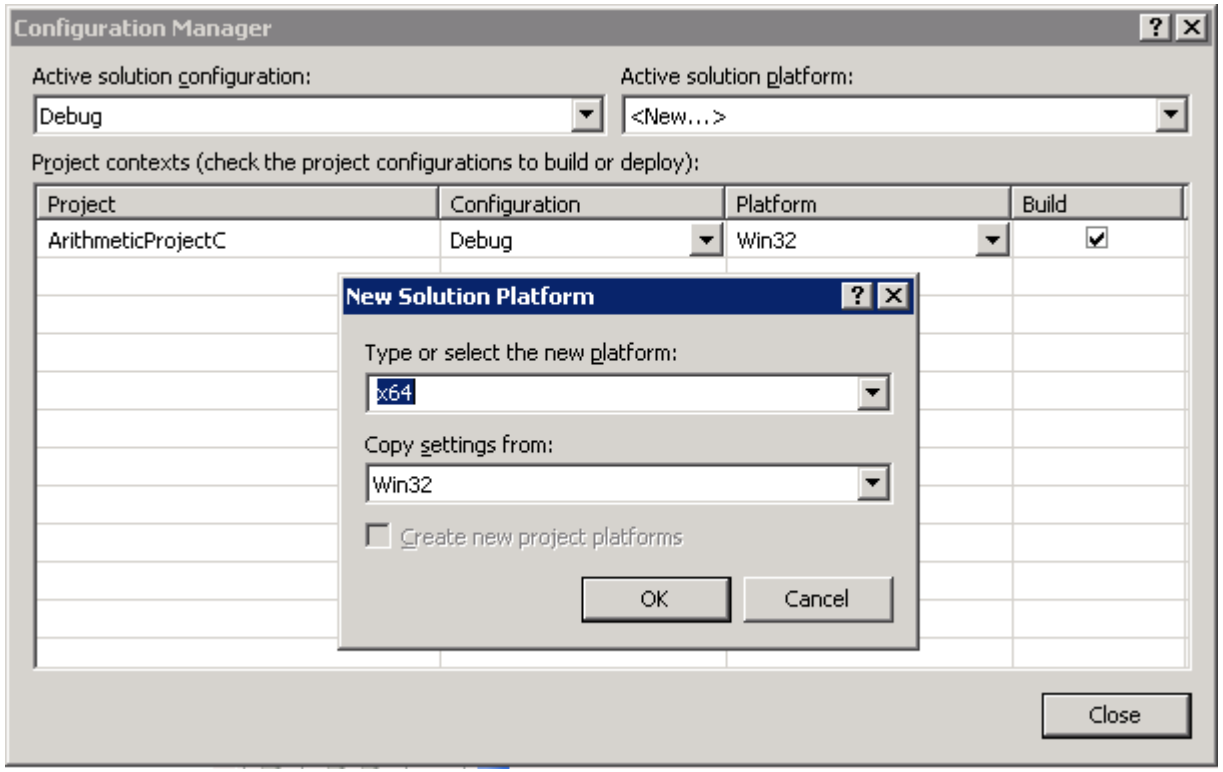
```
; ml64 /Zi ArithmeticProject.asm /link /entry:main /SUBSYSTEM:CONSOLE

_data SEGMENT

a   QWORD 0
b   QWORD 0

_data ENDS

_text SEGMENT

main PROC

    mov             [a], 1
    mov             [b], 1
    mov             rax, [a]
    add             [b], rax

    inc             rax
    mov             [a], rax

    imul            [b]
    mov             [b], rax

    ret

main ENDP

_text ENDS

END
```

Equivalent C program

```
// Visual C++.2005 on AMD x64 platform

__int64 a, b;

int main(int argc, char* argv[])
{
   a = 1;                  // mov [a], 1
   b = 1;                  // mov [b], 1

   b = b + a;              // mov rax, [a]
                           // add [b], rax
   ++a;                    // inc rax
                           // mov [a], rax
   b = a * b;              // imul [b]
                           // mov [b], rax

   // results: [a] = 2 and [b] = 4

   return 0;
}
```

Adding x64 solution platform in Visual C++.2005

- By default the platform is Win32 and your project will be compiled and linked as 32-bit executable. To compile and link it as 64-bit executable you need to create an x64 solution platform in Configuration Manager



WinDbg disassembly output – Debug executable

```
0:000> u main
ArithmeticProjectC!main [c:\dmitri\training-
64\part2\arithmeticprojectc\arithmeticprojectc\arithmeticprojectc.cpp @ 6]:
00000000`00401010 4889542410                    mov  [rsp+0x10],rdx
00000000`00401015 894c2408                      mov  [rsp+0x8],ecx
00000000`00401019 57                            push rdi
00000000`0040101a 48c705a361000001000000        mov  qword ptr [ArithmeticProjectC!a (00000000004071c8)],0x1
00000000`00401025 48c705906100000100000         mov  qword ptr [ArithmeticProjectC!b (00000000004071c0)],0x1
00000000`00401030 488b0591610000                mov  rax,[ArithmeticProjectC!a (00000000004071c8)]
00000000`00401037 488b0d82610000                mov  rcx,[ArithmeticProjectC!b (00000000004071c0)]
00000000`0040103e 4803c8                        add  rcx,rax
00000000`00401041 488bc1                        mov  rax,rcx
00000000`00401044 48890575610000                mov  [ArithmeticProjectC!b (00000000004071c0)],rax
00000000`0040104b 488b0576610000                mov  rax,[ArithmeticProjectC!a (00000000004071c8)]
00000000`00401052 4883c001                      add  rax,0x1
00000000`00401056 4889056b610000                mov  [ArithmeticProjectC!a (00000000004071c8)],rax
00000000`0040105d 488b0564610000                mov  rax,[ArithmeticProjectC!a (00000000004071c8)]
00000000`00401064 480faf0554610000              imul rax,[ArithmeticProjectC!b (00000000004071c0)]
00000000`0040106c 4889054d610000                mov  [ArithmeticProjectC!b (00000000004071c0)],rax
00000000`00401073 33c0                          xor  eax,eax
00000000`00401075 5f                            pop  rdi
```

```
mov [a], 1     ; [a] := 1
mov [b], 1     ; [b] := 1
mov rax, [a]   ; [b] := [b] + [a]
mov rcx, [b]   ;
add rcx, rax   ;
mov rax, rcx   ;
mov [b], rax   ;
mov rax, [a]   ; [a] := [a] + 1
add rax, 1     ;
mov [a], rax   ;
mov rax, [a]   ; [b] := [b] * [a]
imul rax, [b]  ;
mov [b], rax   ;
```

WinDbg disassembly output – Release executable

```
0:000> u main
ArithmeticProjectC!main [c:\dmitri\training-
    64\part2\arithmeticprojectc\arithmeticprojectc\arithmeticprojectc.cpp @ 6]:
00000000`00401000 48c705cd25000002000000 mov qword ptr [ArithmeticProjectC!a (00000000004035d8)],0x2
00000000`0040100b 48c705ba25000004000000 mov qword ptr [ArithmeticProjectC!b (00000000004035d0)],0x4
00000000`00401016 33c0                   xor     eax,eax
00000000`00401018 c3                     ret
```

```
mov [a], 2   ; [a] := 2
mov [b], 4   ; [b] := 4
```

Numbers and their representations

- number of stones 
- a person can only count up to 3 

The last picture is representation (notation) of the number of stones

Decimal representation (base 10)

- 12 stones

$12_{dec} = 1 * 10 + 2$ or $1*10^1 + 2*10^0$

- 123 stones

$123_{dec} = 1 * 100 + 2 * 10 + 3$ or $1 * 10^2 + 2 * 10^1 + 3*10^0$

$N_{dec} = a_n*10^n + a_{n-1}*10^{n-2} + \ldots + a_2*10^2 + a_1*10^1 + a_0*10^0$ $0 <= a_i <= 9$

$$N_{dec} = \sum_{i=0}^{n} a_i*10^i$$

Ternary representation (base 3)

- 12 stones

110 in ternary representation (notation)

$12_{dec} = 1*3^2 + 1*3^1 + 0*3^0$

$N_{dec} = a_n*3^n + a_{n-1}*3^{n-2} + \ldots + a_2*3^2 + a_1*3^1 + a_0*3^0$ $\quad a_i = 0$ or 1 or 2

$$N_{dec} = \sum_{i=0}^{n} a_i*3^i$$

Binary representation (base 2)

- 12 stones

1100 in binary representation (notation)

$12_{dec} = 1*2^3 + 1*2^2 + 0*2^1 + 0*2^0$

$N_{dec} = a_n*2^n + a_{n-1}*2^{n-2} + \ldots + a_2*2^2 + a_1*2^1 + a_0*2^0$ $\quad a_i = 0$ or 1

$$N_{dec} = \sum_{i=0}^{n} a_i*2^i$$

Hexadecimal representation (base 16)

- 12 stones

12_{dec} = C in hexadecimal representation (notation)

- 123 stones

123_{dec} = 7B in hexadecimal representation (notation)

$123_{dec} = 7*16^1 + 11*16^0$ …8, 9, A, B, C, D, E, F

$$N_{dec} = \sum_{i=0}^{n} a_i * 16^i$$

Why hexadecimal is used?

- 110001010011 (binary notation)

$3155_{dec} = 1*2^{11} + 1*2^{10} + 0*2^9 + 0*2^8 + 0*2^7 + 1*2^6 + 0*2^5 + 1*2^4 + 0*2^3 + 0*2^2 + 1*2^1 + 1*2^0$

1100*0101***0011** is **C53** in hexadecimal

12$_{dec}$ *5*$_{dec}$ ***3***$_{dec}$

C$_{hex}$ *5*$_{hex}$ ***3***$_{hex}$

- In WinDbg memory addresses are displayed in hexadecimal notation

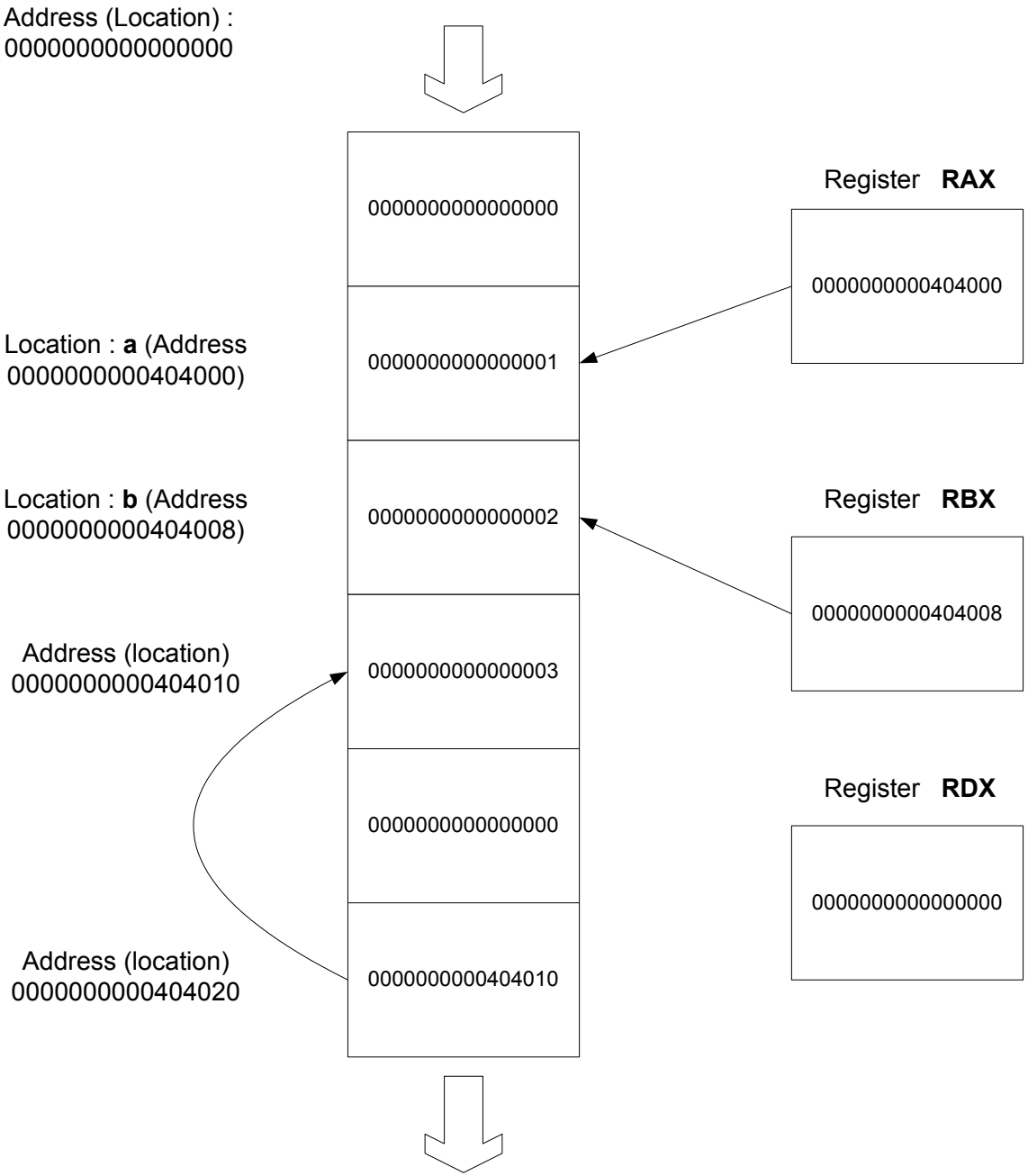
Binary <-> Decimal <-> Hexadecimal

0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

Pointers

- Pointer is a memory cell or a register that contains an address of another memory cell. Has its own address (as any memory cell)
- Another name: Indirect address (vs. direct address, the address of memory cell). Another level of indirection
- Levels of indirection: pointer to a pointer

  (Memory cell or register that contains the address of another memory cell that contains the address of another memory cell)


Address (Location) :
0000000000000000
0000000000000000
Register RAX
0000000000404000
Location : a (Address
0000000000404000)
0000000000000001
Location : b (Address
0000000000404008)
0000000000000002
Register RBX
0000000000404008
Address (location)
0000000000404010
0000000000000003
Register RDX
0000000000000000
0000000000000000
Address (location)
0000000000404020
0000000000404010

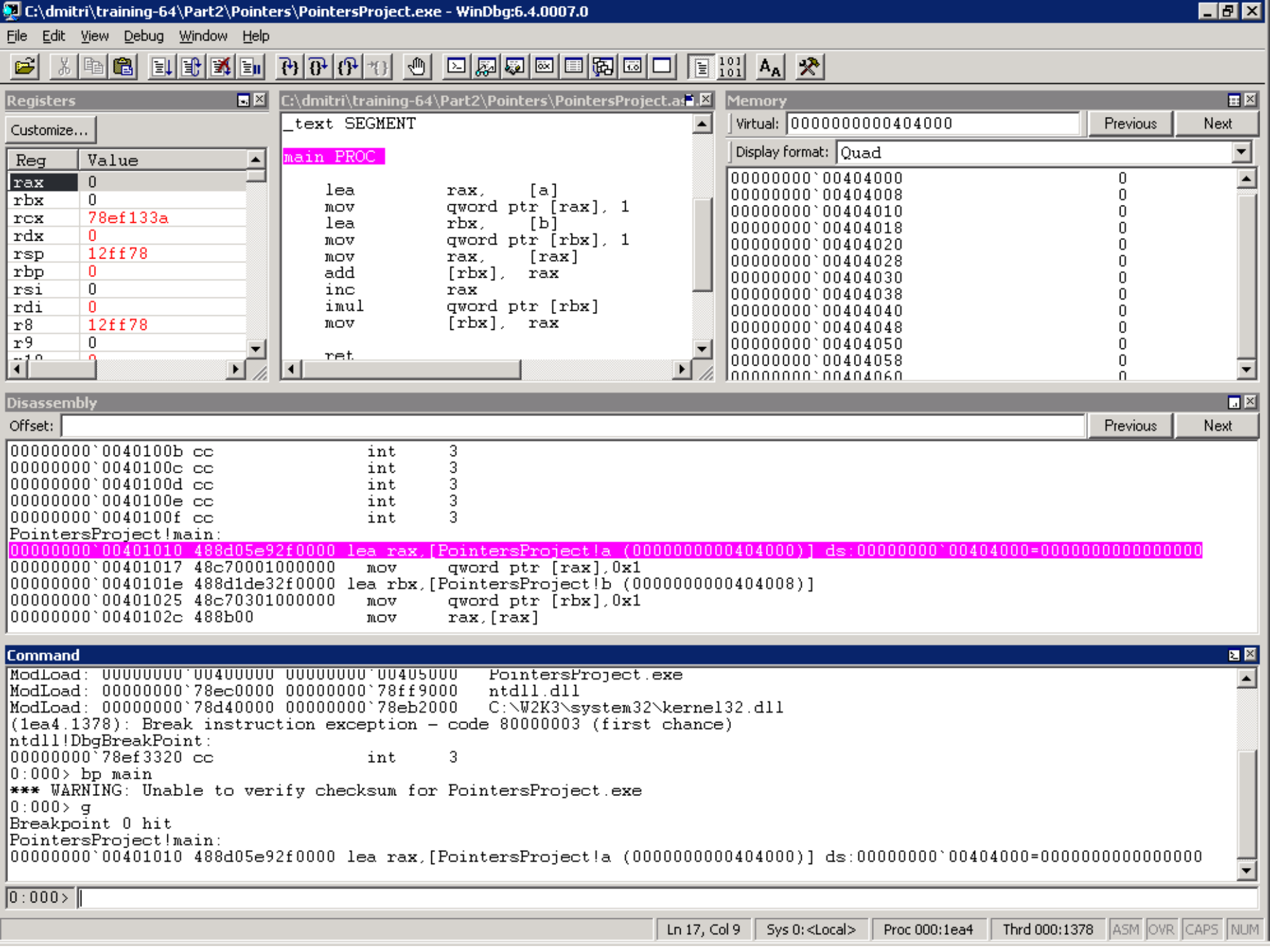
Pointers Project – Memory Layout and Registers

- Two memory addresses (locations):
  "a" and "b". We can think about "a" and "b" as names of addresses (locations)
- Notation [a] means contents at the memory address (location) "a"
- Registers *RAX* and *RBX;* pointers to "a" and "b"; contain addresses of "a" and "b"
- Notation [RAX] means contents of the memory cell whose address is in RAX
- In Visual C++ we declare pointers to 64-bit "a" and "b" as:

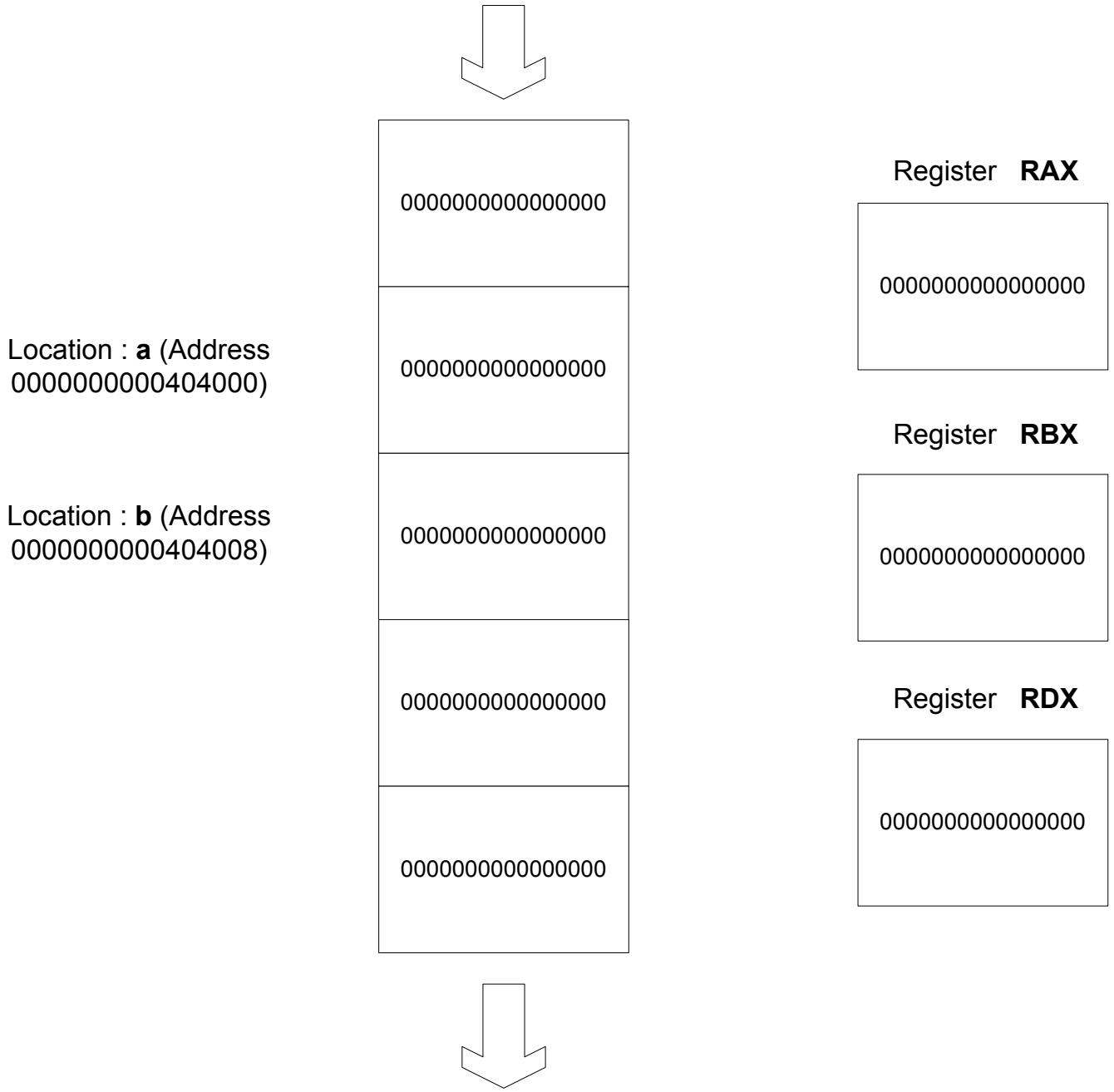
__int64 *a, *b;

PointersProject.asm

```
; ml64 /Zi PointersProject.asm /link /entry:main /SUBSYSTEM:CONSOLE

_data SEGMENT

a     QWORD           0
b     QWORD           0

_data ENDS

_text SEGMENT

main PROC

    lea         rax,   [a]
    mov         qword ptr [rax], 1
    lea         rbx,   [b]
    mov         qword ptr [rbx], 1
    mov         rax,   [rax]
    add         [rbx], rax
    inc         rax
    imul        qword ptr [rbx]
    mov         [rbx],  rax

    ret

main ENDP

_text ENDS

END
```


C:\dmitri\training-64\Part2\Pointers\PointersProject.exe - WinDbg:6.4.0007.0
File Edit View Debug Window Help
Registers
Customize...
Reg Value
rax 0
rbx 0
rcx 78ef133a
rdx 0
rsp 12ff78
rbp 0
rsi 0
rdi 0
r8 12ff78
r9 0
C:\dmitri\training-64\Part2\Pointers\PointersProject.as
_text SEGMENT
main PROC
lea rax, [a]
mov qword ptr [rax], 1
lea rbx, [b]
mov qword ptr [rbx], 1
mov rax, [rax]
add [rbx], rax
inc rax
imul qword ptr [rbx]
mov [rbx], rax
ret
Memory
Virtual: 0000000000404000
Previous
Next
Display format: Quad
00000000`00404000 0
00000000`00404008 0
00000000`00404010 0
00000000`00404018 0
00000000`00404020 0
00000000`00404028 0
00000000`00404030 0
00000000`00404038 0
00000000`00404040 0
00000000`00404048 0
00000000`00404050 0
00000000`00404058 0
Disassembly
Offset:
Previous
Next
00000000`0040100b cc int 3
00000000`0040100c cc int 3
00000000`0040100d cc int 3
00000000`0040100e cc int 3
00000000`0040100f cc int 3
PointersProject!main:
00000000`00401010 488d05e92f0000 lea rax,[PointersProject!a (0000000000404000)] ds:00000000`00404000=0000000000000000
00000000`00401017 48c70001000000 mov qword ptr [rax],0x1
00000000`0040101e 488d1de32f0000 lea rbx,[PointersProject!b (0000000000404008)]
00000000`00401025 48c70301000000 mov qword ptr [rbx],0x1
00000000`0040102c 488b00 mov rax,[rax]
Command
ModLoad: 00000000`00400000 00000000`00405000 PointersProject.exe
ModLoad: 00000000`78ec0000 00000000`78ff9000 ntdll.dll
ModLoad: 00000000`78d40000 00000000`78eb2000 C:\W2K3\system32\kernel32.dll
(1ea4.1378): Break instruction exception - code 80000003 (first chance)
ntdll!DbgBreakPoint:
00000000`78ef3320 cc int 3
0:000> bp main
*** WARNING: Unable to verify checksum for PointersProject.exe
0:000> g
Breakpoint 0 hit
PointersProject!main:
00000000`00401010 488d05e92f0000 lea rax,[PointersProject!a (0000000000404000)] ds:00000000`00404000=0000000000000000
0:000>
Ln 17, Col 9
Sys 0:<Local>
Proc 000:1ea4
Thrd 000:1378
ASM
OVR
CAPS
NUM

Picture 1


0000000000000000
0000000000000000
0000000000000000
0000000000000000
0000000000000000
Location : **a** (Address 0000000000404000)
Location : **b** (Address 0000000000404008)
Register **RAX**
0000000000000000
Register **RBX**
0000000000000000
Register **RDX**
0000000000000000

Pointers Project - Calculations

```
rax := address a
[rax] := 1 ; [a] = 1
rbx := address b
[rbx] := 1 ; [b] = 1
[rbx] := [rbx] + [rax]
            ; [b] = 2
[rbx] := [rbx] * 2
            ; [b] = 4
```

Using pointers to assign numbers to memory cells

- rax := address a
- [rax] := 1

means using the contents of rax as address of a memory cell and assign a value to the contents of this memory cell

- In C language it is called a “pointer” and we write:

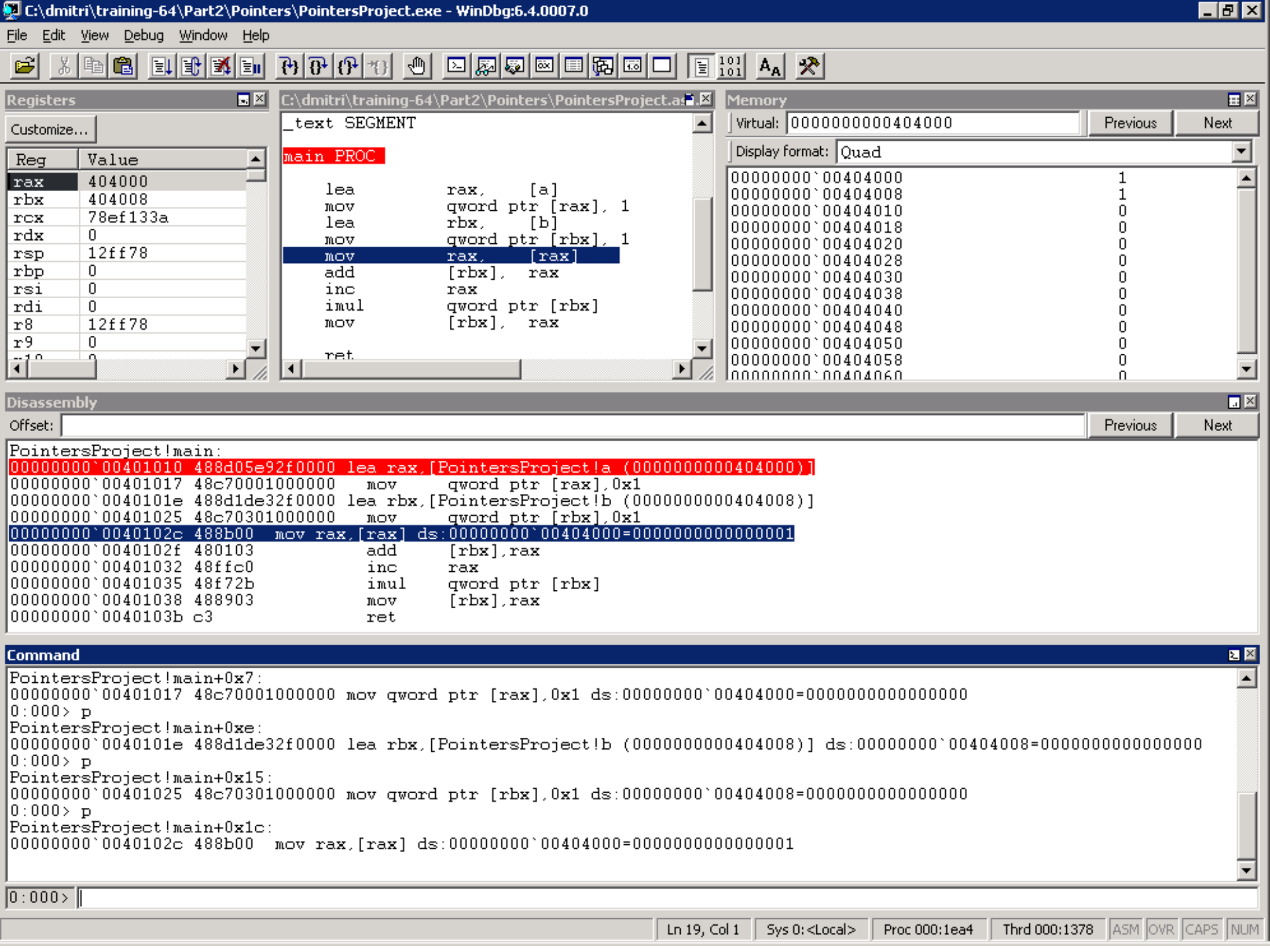
```
__int64 *pa = &a;  // definition of a pointer
*pa = 1;           // get memory cell (dereferencing a pointer)
                   // and assign a value to it
```

- In Assembler we write:

```
lea rax, [a]                ; load the address a into rax
mov qword ptr [rax], 1      ; use rax as a pointer
```

- In WinDbg disassembly output we see:

```
00000000`00401010 488d05e92f0000 lea rax,[PointersProject!a (0000000000404000)]
00000000`00401017 48c70001000000 mov qword ptr [rax],0x1
```

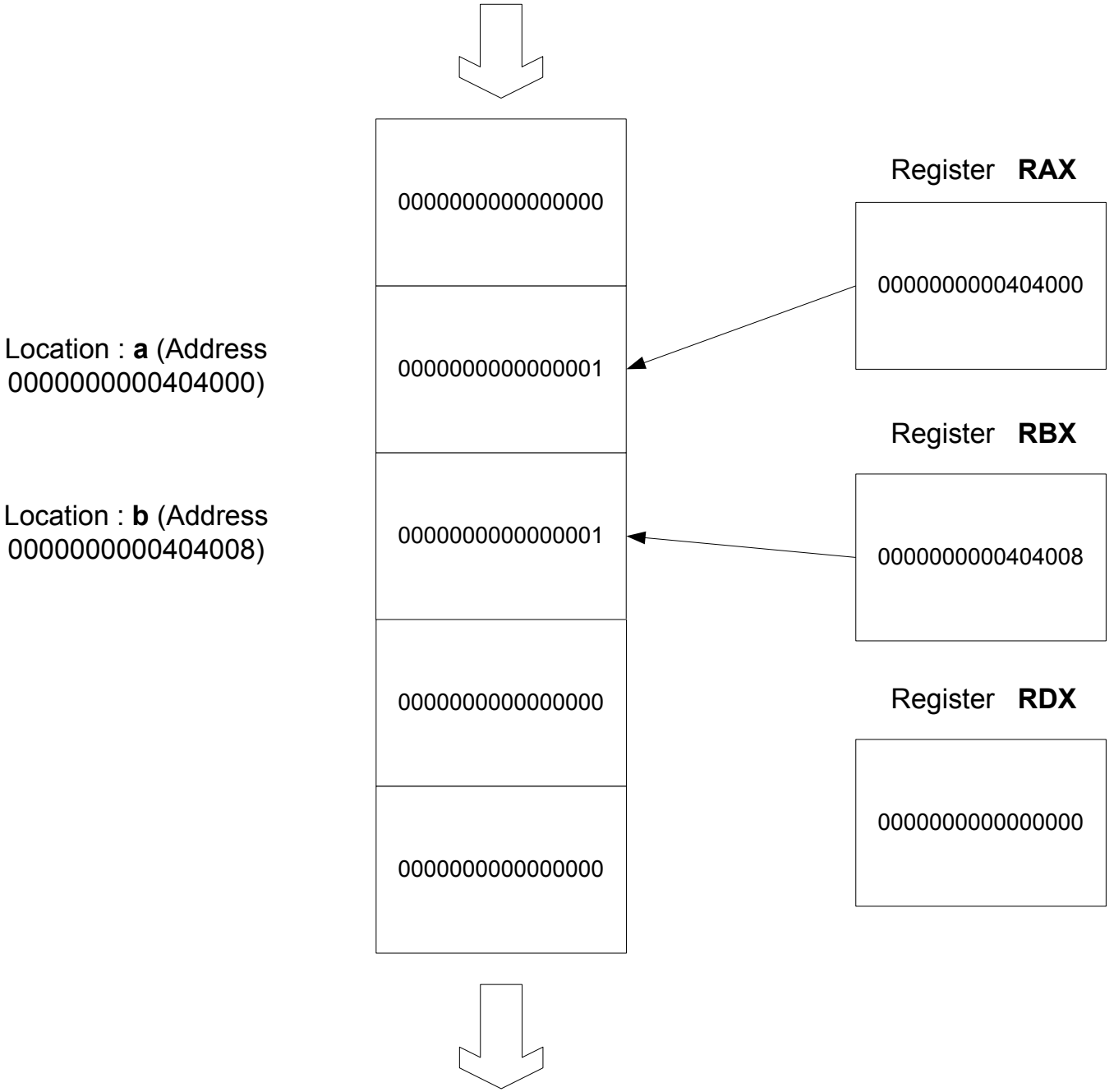

C:\dmitri\training-64\Part2\Pointers\PointersProject.exe - WinDbg:6.4.0007.0
File Edit View Debug Window Help
Registers
Customize...
Reg Value
rax 404000
rbx 404008
rcx 78ef133a
rdx 0
rsp 12ff78
rbp 0
rsi 0
rdi 0
r8 12ff78
r9 0
C:\dmitri\training-64\Part2\Pointers\PointersProject.as
_text SEGMENT
main PROC
lea rax, [a]
mov qword ptr [rax], 1
lea rbx, [b]
mov qword ptr [rbx], 1
mov rax, [rax]
add [rbx], rax
inc rax
imul qword ptr [rbx]
mov [rbx], rax
ret
Memory
Virtual: 0000000000404000
Previous
Next
Display format: Quad
00000000`00404000 1
00000000`00404008 1
00000000`00404010 0
00000000`00404018 0
00000000`00404020 0
00000000`00404028 0
00000000`00404030 0
00000000`00404038 0
00000000`00404040 0
00000000`00404048 0
00000000`00404050 0
00000000`00404058 0
Disassembly
Offset:
Previous
Next
PointersProject!main:
00000000`00401010 488d05e92f0000 lea rax,[PointersProject!a (0000000000404000)]
00000000`00401017 48c70001000000 mov qword ptr [rax],0x1
00000000`0040101e 488d1de32f0000 lea rbx,[PointersProject!b (0000000000404008)]
00000000`00401025 48c70301000000 mov qword ptr [rbx],0x1
00000000`0040102c 488b00 mov rax,[rax] ds:00000000`00404000=0000000000000001
00000000`0040102f 480103 add [rbx],rax
00000000`00401032 48ffc0 inc rax
00000000`00401035 48f72b imul qword ptr [rbx]
00000000`00401038 488903 mov [rbx],rax
00000000`0040103b c3 ret
Command
PointersProject!main+0x7:
00000000`00401017 48c70001000000 mov qword ptr [rax],0x1 ds:00000000`00404000=0000000000000000
0:000> p
PointersProject!main+0xe:
00000000`0040101e 488d1de32f0000 lea rbx,[PointersProject!b (0000000000404008)] ds:00000000`00404008=0000000000000000
0:000> p
PointersProject!main+0x15:
00000000`00401025 48c70301000000 mov qword ptr [rbx],0x1 ds:00000000`00404008=0000000000000000
0:000> p
PointersProject!main+0x1c:
00000000`0040102c 488b00 mov rax,[rax] ds:00000000`00404000=0000000000000001
0:000>
Ln 19, Col 1
Sys 0:<Local>
Proc 000:1ea4
Thrd 000:1378
ASM OVR CAPS NUM

Pointers Project - Calculations

```
rax := address a
[rax] := 1  ; [a] = 1
rbx := address b
[rbx] := 1  ; [b] = 1
[rbx] := [rbx] + [rax]
            ; [b] = 2
[rbx] := [rbx] * 2
            ; [b] = 4
```

```
lea rax, [a]
mov qword ptr [rax], 1
lea rbx, [b]
mov qword ptr [rbx], 1
```

Picture 2


0000000000000000
0000000000000001
0000000000000001
0000000000000000
0000000000000000
Location : **a** (Address 0000000000404000)
Location : **b** (Address 0000000000404008)
Register **RAX**
0000000000404000
Register **RBX**
0000000000404008
Register **RDX**
0000000000000000

Adding numbers using pointers

- [rbx] := [rbx] + [rax]

[rax] and [rbx] mean contents of memory cells whose addresses (locations) are stored in rax and rbx

- In C language we write:

***pb = *pb + *pa;**

- In Assembler we use instruction **add**

We cannot use both memory addresses in one step (instruction):

add [rbx], [rax]

We can only use **add [rbx], register**

register := [rax]

[rbx] := [rbx] + register

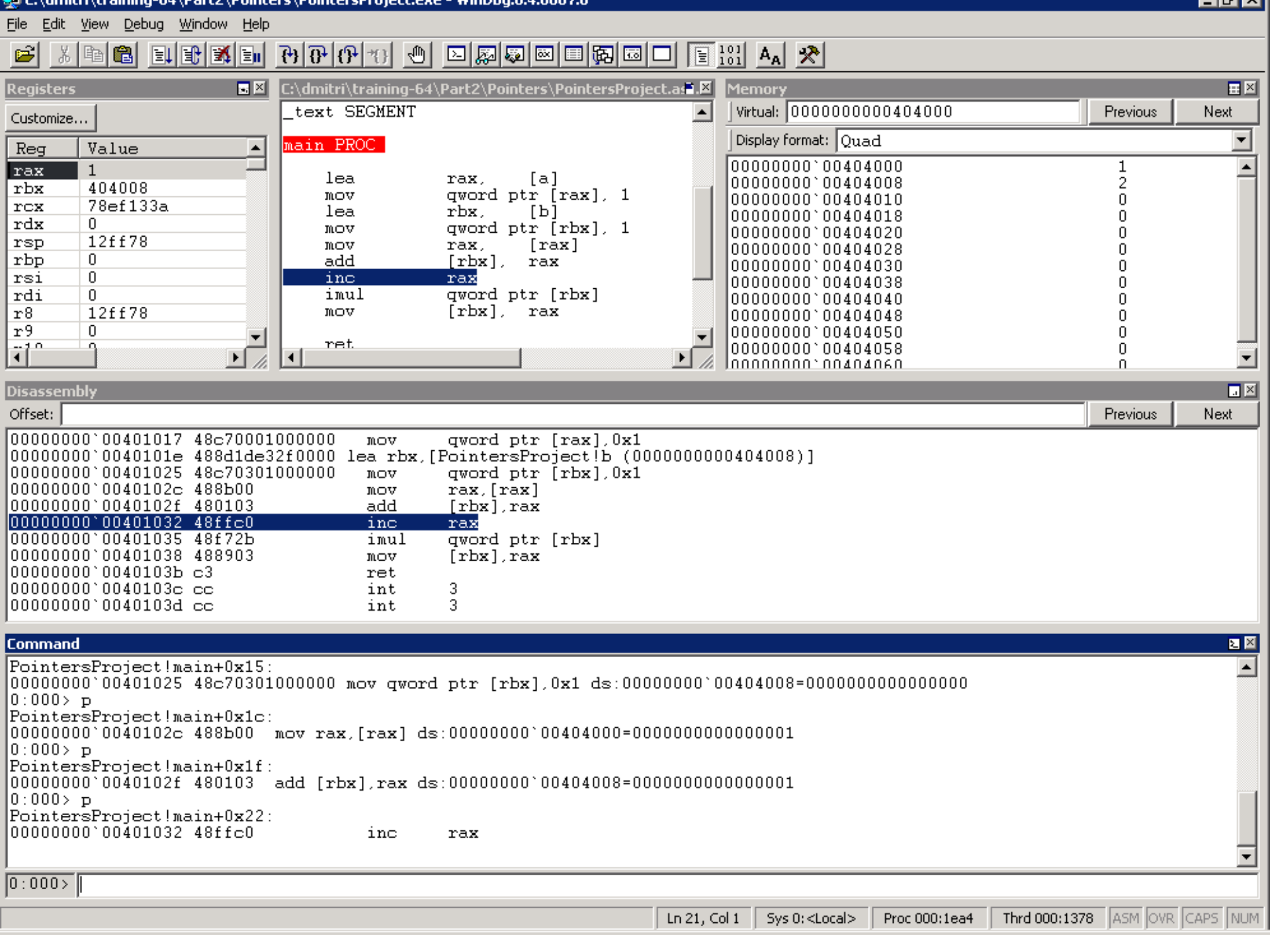
- In Assembler we write:

mov rax, [rax]

add [rbx], rax

- In WinDbg disassembly output we see:

```
00000000`0040102c 488b00  mov rax,[rax]
00000000`0040102f 480103  add [rbx],rax
```

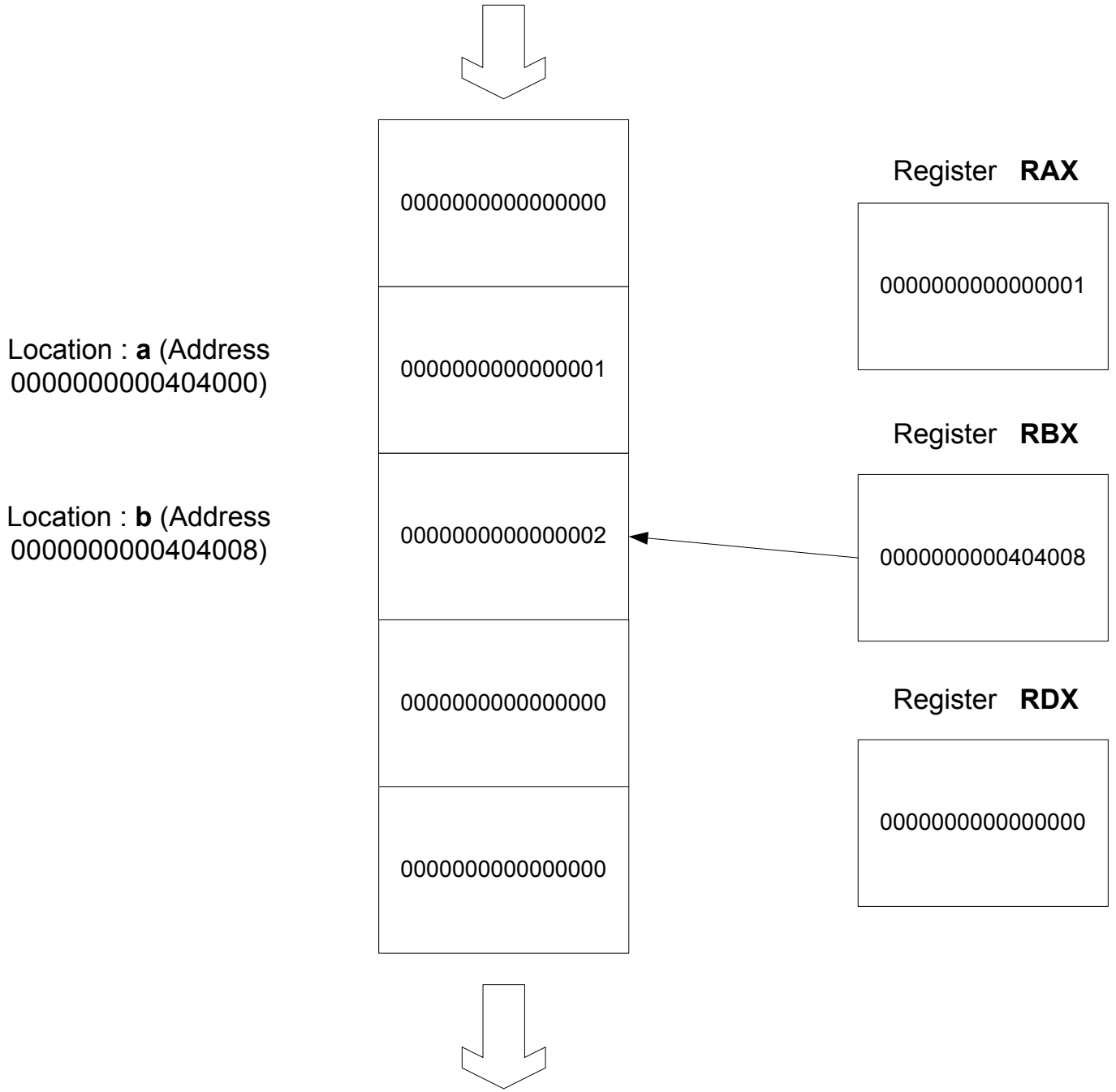

File Edit View Debug Window Help
Registers
Customize...
Reg Value
rax 1
rbx 404008
rcx 78ef133a
rdx 0
rsp 12ff78
rbp 0
rsi 0
rdi 0
r8 12ff78
r9 0
C:\dmitri\training-64\Part2\Pointers\PointersProject.as
_text SEGMENT
main PROC
lea rax, [a]
mov qword ptr [rax], 1
lea rbx, [b]
mov qword ptr [rbx], 1
mov rax, [rax]
add [rbx], rax
inc rax
imul qword ptr [rbx]
mov [rbx], rax
ret
Memory
Virtual: 0000000000404000
Previous
Next
Display format: Quad
00000000`00404000 1
00000000`00404008 2
00000000`00404010 0
00000000`00404018 0
00000000`00404020 0
00000000`00404028 0
00000000`00404030 0
00000000`00404038 0
00000000`00404040 0
00000000`00404048 0
00000000`00404050 0
00000000`00404058 0
Disassembly
Offset:
Previous
Next
00000000`00401017 48c70001000000 mov qword ptr [rax],0x1
00000000`0040101e 488d1de32f0000 lea rbx,[PointersProject!b (0000000000404008)]
00000000`00401025 48c70301000000 mov qword ptr [rbx],0x1
00000000`0040102c 488b00 mov rax,[rax]
00000000`0040102f 480103 add [rbx],rax
00000000`00401032 48ffc0 inc rax
00000000`00401035 48f72b imul qword ptr [rbx]
00000000`00401038 488903 mov [rbx],rax
00000000`0040103b c3 ret
00000000`0040103c cc int 3
00000000`0040103d cc int 3
Command
PointersProject!main+0x15:
00000000`00401025 48c70301000000 mov qword ptr [rbx],0x1 ds:00000000`00404008=0000000000000000
0:000> p
PointersProject!main+0x1c:
00000000`0040102c 488b00 mov rax,[rax] ds:00000000`00404000=0000000000000001
0:000> p
PointersProject!main+0x1f:
00000000`0040102f 480103 add [rbx],rax ds:00000000`00404008=0000000000000001
0:000> p
PointersProject!main+0x22:
00000000`00401032 48ffc0 inc rax
0:000>
Ln 21, Col 1
Sys 0:<Local>
Proc 000:1ea4
Thrd 000:1378
ASM
OVR
CAPS
NUM

Pointers Project - Calculations

```
rax := address a
[rax] := 1   ; [a] = 1
rbx := address b
[rbx] := 1   ; [b] = 1
[rbx] := [rbx] + [rax]
             ; [b] = 2
[rbx] := [rbx] * 2
             ; [b] = 4
```

```
lea rax,   [a]
mov qword ptr [rax], 1
lea rbx,   [b]
mov qword ptr [rbx], 1
mov rax,   [rax]
add [rbx], rax
```

Picture 3


0000000000000000
0000000000000001
0000000000000002
0000000000000000
0000000000000000
Location : **a** (Address 0000000000404000)
Location : **b** (Address 0000000000404008)
Register **RAX**
0000000000000001
Register **RBX**
0000000000404008
Register **RDX**
0000000000000000

Multiplying numbers using pointers

- [rbx] := [rbx] * 2

Means multiply contents of the memory whose address is stored in rbx by 2

- In C language we write:

***pb = *pb * 2;** or ***pb *= 2;**

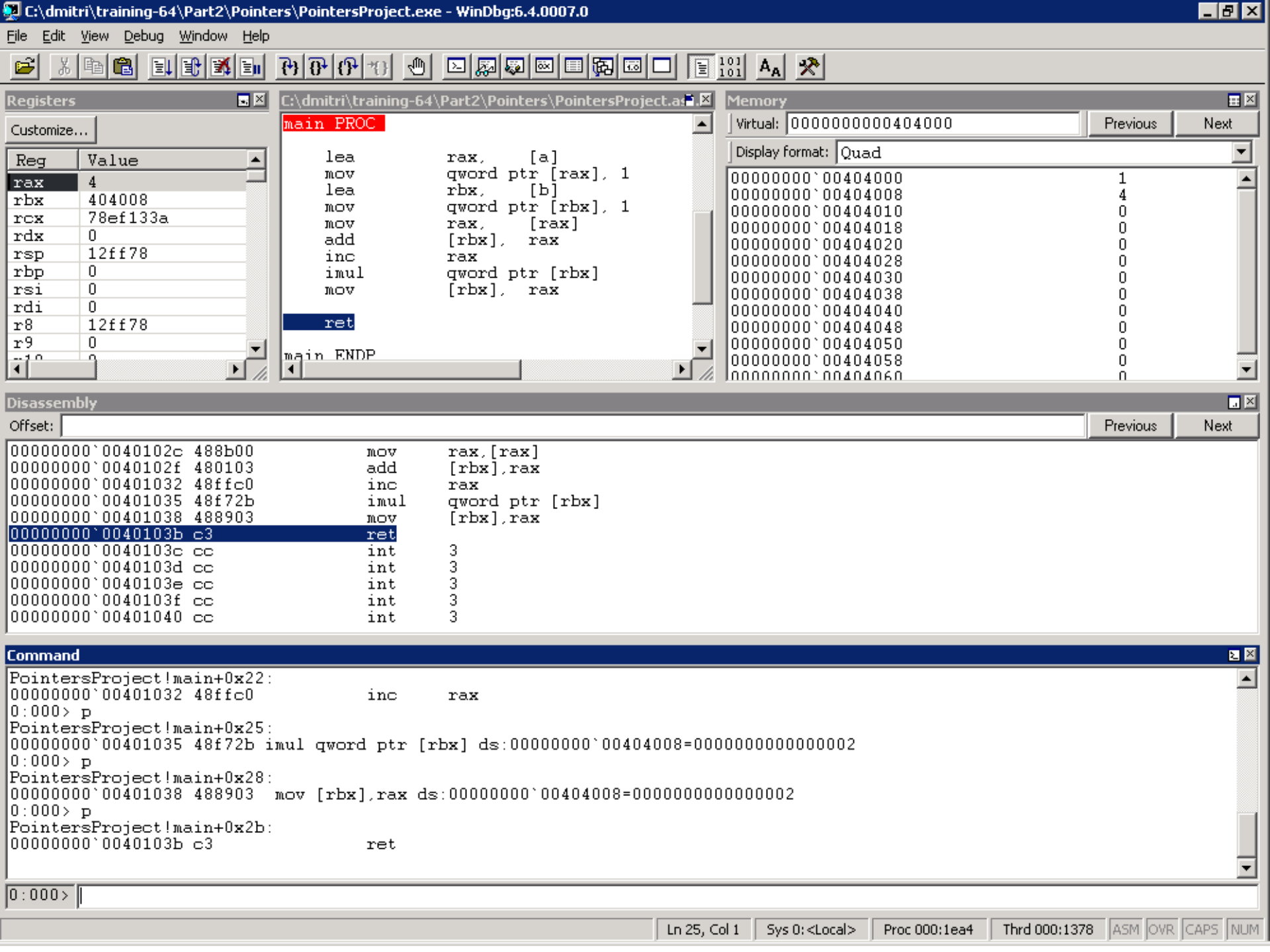
- In Assembler we use instruction **imul** (integer multiply)

imul qword ptr [rbx]

Means [rbx] := [rbx] * rax, so we have to put 2 into rax, but we already have 1 in **rax** so we use **inc rax** before **imul** to increment by 1

- In WinDbg disassembly output we see:

```
00000000`00401032 48ffc0    inc     rax
00000000`00401035 48f72b    imul    qword ptr [rbx]
00000000`00401038 488903    mov     [rbx],rax
```

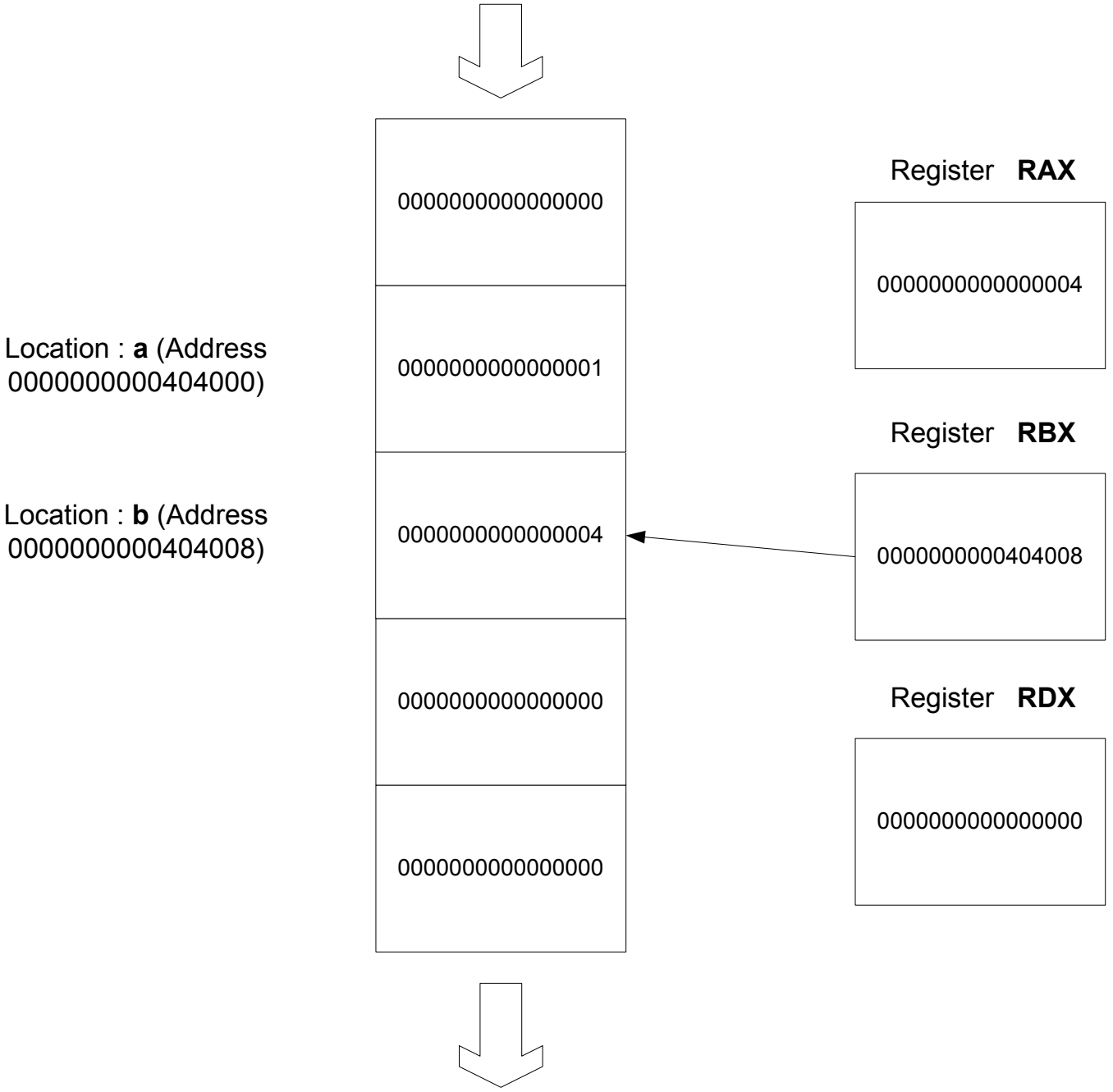

C:\dmitri\training-64\Part2\Pointers\PointersProject.exe - WinDbg:6.4.0007.0
File Edit View Debug Window Help
Registers
Customize...
Reg Value
rax 4
rbx 404008
rcx 78ef133a
rdx 0
rsp 12ff78
rbp 0
rsi 0
rdi 0
r8 12ff78
r9 0
C:\dmitri\training-64\Part2\Pointers\PointersProject.as
main PROC
lea rax, [a]
mov qword ptr [rax], 1
lea rbx, [b]
mov qword ptr [rbx], 1
mov rax, [rax]
add [rbx], rax
inc rax
imul qword ptr [rbx]
mov [rbx], rax
ret
main ENDP
Memory
Virtual: 0000000000404000
Previous
Next
Display format: Quad
00000000`00404000 1
00000000`00404008 4
00000000`00404010 0
00000000`00404018 0
00000000`00404020 0
00000000`00404028 0
00000000`00404030 0
00000000`00404038 0
00000000`00404040 0
00000000`00404048 0
00000000`00404050 0
00000000`00404058 0
Disassembly
Offset:
Previous
Next
00000000`0040102c 488b00 mov rax,[rax]
00000000`0040102f 480103 add [rbx],rax
00000000`00401032 48ffc0 inc rax
00000000`00401035 48f72b imul qword ptr [rbx]
00000000`00401038 488903 mov [rbx],rax
00000000`0040103b c3 ret
00000000`0040103c cc int 3
00000000`0040103d cc int 3
00000000`0040103e cc int 3
00000000`0040103f cc int 3
00000000`00401040 cc int 3
Command
PointersProject!main+0x22:
00000000`00401032 48ffc0 inc rax
0:000> p
PointersProject!main+0x25:
00000000`00401035 48f72b imul qword ptr [rbx] ds:00000000`00404008=0000000000000002
0:000> p
PointersProject!main+0x28:
00000000`00401038 488903 mov [rbx],rax ds:00000000`00404008=0000000000000002
0:000> p
PointersProject!main+0x2b:
00000000`0040103b c3 ret
0:000>
Ln 25, Col 1
Sys 0:<Local>
Proc 000:1ea4
Thrd 000:1378
ASM OVR CAPS NUM

Pointers Project - Calculations

```
rax := address a
[rax] := 1   ; [a] = 1
rbx := address b
[rbx] := 1   ; [b] = 1
[rbx] := [rbx] + [rax]
             ; [b] = 2
[rbx] := [rbx] * 2
             ; [b] = 4
```

```
lea  rax,   [a]
mov  qword ptr [rax], 1
lea  rbx,   [b]
mov  qword ptr [rbx], 1
mov  rax,   [rax]
add  [rbx], rax
inc  rax
imul [rbx]
mov  [rbx], rax
```

Picture 4


0000000000000000
0000000000000001
0000000000000004
0000000000000000
0000000000000000
Location : **a** (Address 0000000000404000)
Location : **b** (Address 0000000000404008)
Register **RAX**
0000000000000004
Register **RBX**
0000000000404008
Register **RDX**
0000000000000000

What's next?

- More practice with binary and hexadecimal notations.
- Bits, bytes, words, double and quad words.
- Pointers to bytes, double and quad words.
- Pointers as variables. Null pointer.
- We will rewrite our arithmetic C project using pointers as variables.