Practical Foundations of Debugging for x64 platforms

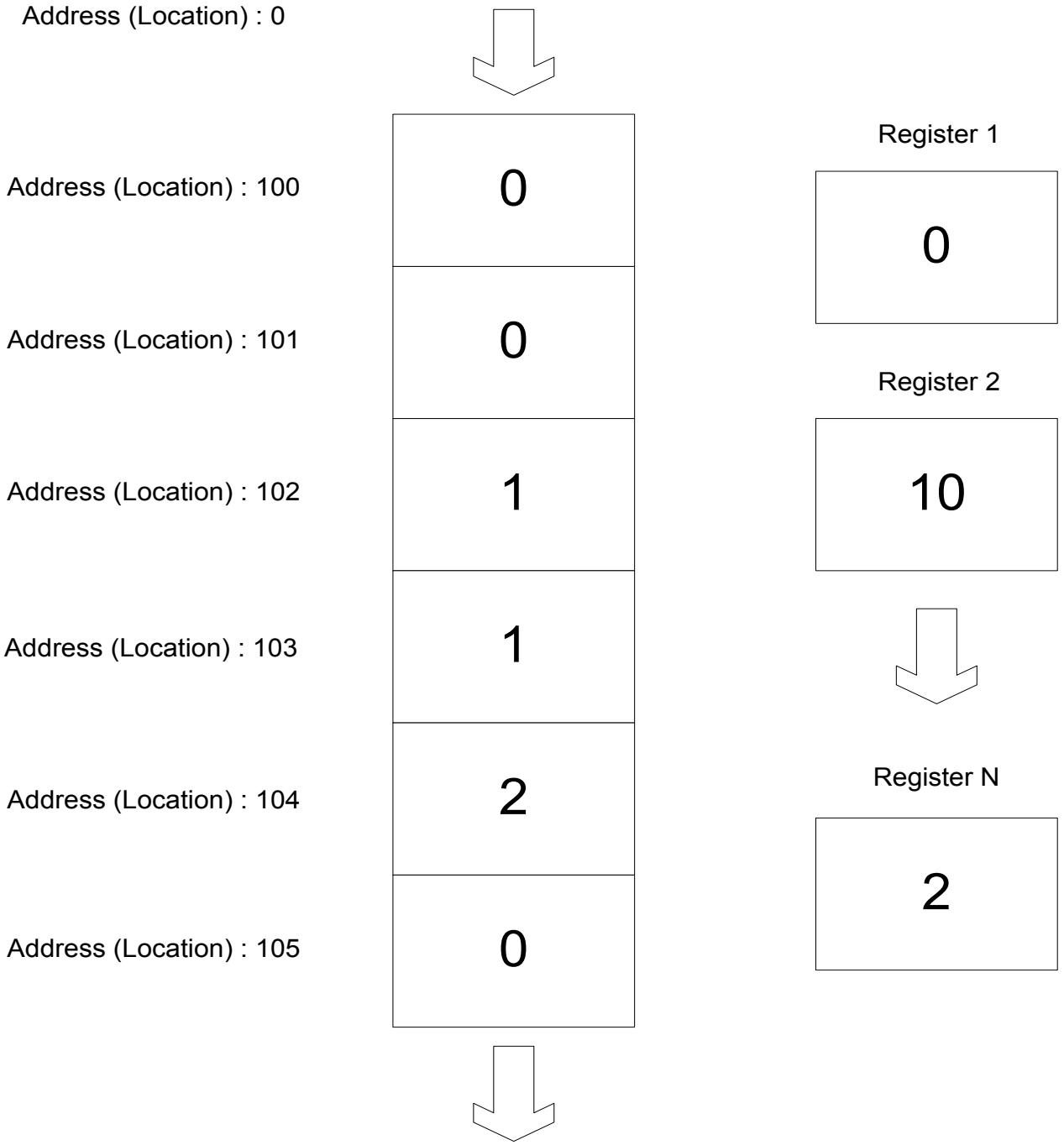
Part 1: Memory, Registers and Simple Arithmetic

Designed, written and illustrated by Dmitry Vostokov

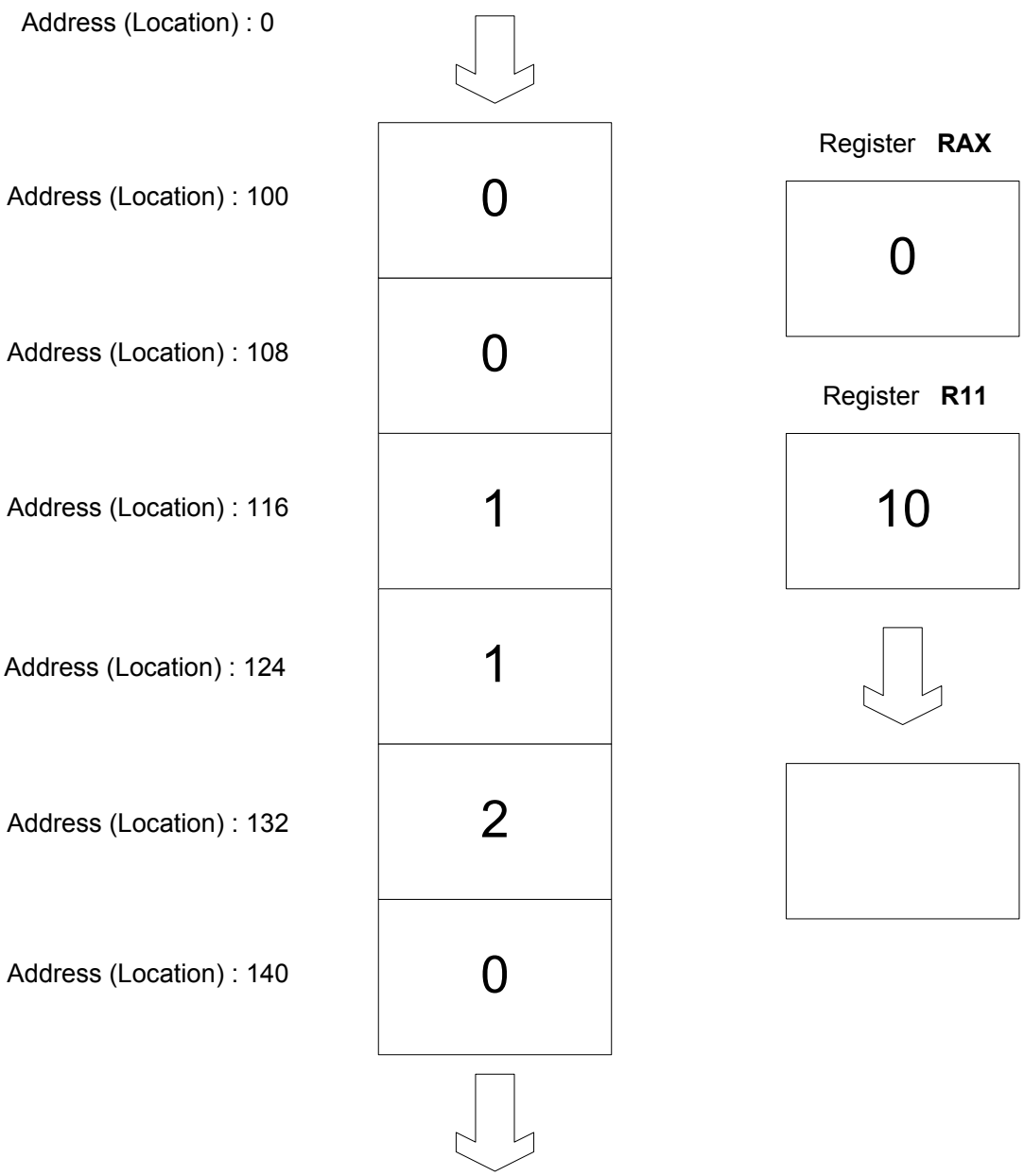
Memory and Registers inside idealized computer

- Computer memory consists of a sequence of memory cells and each cell has a unique address (location). Every cell contains a “number”. We refer to these “numbers” as contents at the given address (location).
- Think of registers as the standalone memory cells. The name of the register is its address.

Picture 1


Address (Location) : 0
Address (Location) : 100
0
Address (Location) : 101
0
Address (Location) : 102
1
Address (Location) : 103
1
Address (Location) : 104
2
Address (Location) : 105
0
Register 1
0
Register 2
10
Register N
2

Memory and 64bit Registers inside x64 PC



Arithmetic 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"

- 64bit Registers *RAX, RDX, …, R10, R11, …*

- In C we define 64bit memory locations "a" and "b" as:

```
static __int64 a, b;
```

- In MASM we define 64bit memory locations as

```
a   QWORD 0
b   QWORD 0
```

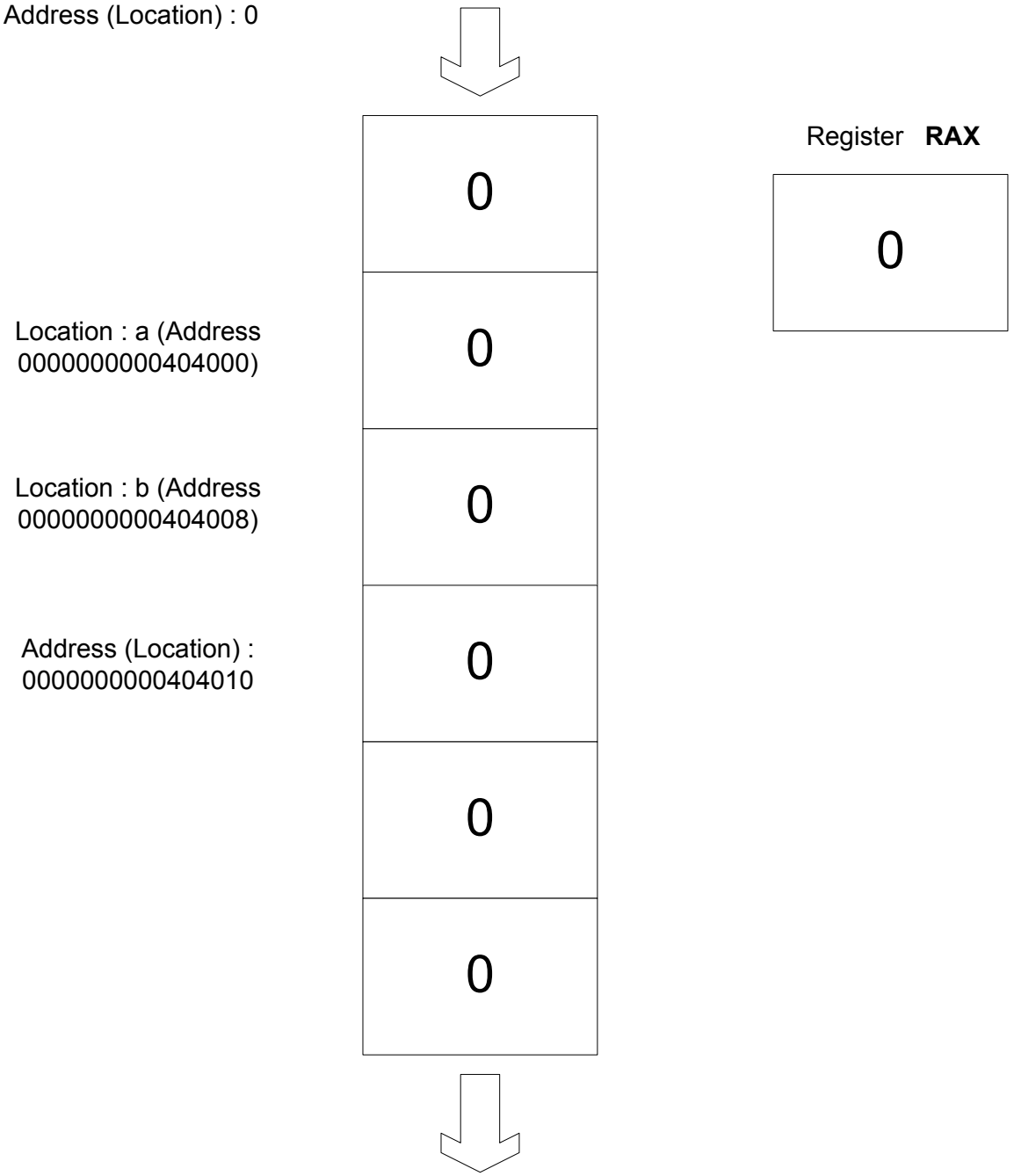
Inline assembler (__asm) is not supported in AMD x64 MS C/C++ compiler so I used MASM64

```
; 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

    imul            [b]
    mov             [b], rax

    ret

main ENDP

_text ENDS

END
```

Loading the executable into WinDbg

```
icrosoft (R) Windows Debugger  Version 6.4.0007.2
Copyright (c) Microsoft Corporation. All rights reserved.

CommandLine: C:\dmitri\training-64\Part1\Project1\ArithmeticProject.exe
Symbol search path is: SRV*c:\websymbols*http://msdl.microsoft.com/download/symbols
Executable search path is:
ModLoad: 00000000`00400000 00000000`00405000   ArithmeticProject.exe
ModLoad: 00000000`78ec0000 00000000`78ff9000   ntdll.dll
ModLoad: 00000000`78d40000 00000000`78eb2000   C:\W2K3\system32\kernel32.dll
(3470.3b54): Break instruction exception - code 80000003 (first chance)
ntdll!DbgBreakPoint:
00000000`78ef3320 cc              int     3

0:000> u main
ArithmeticProject!main [C:\dmitri\training-64\Part1\Project1\ArithmeticProject.asm @ 13]:
00000000`00401010 48c705e52f000001000000 mov qword ptr [ArithmeticProject!a (0000000000404000)],0x1
00000000`0040101b 48c705e22f000001000000 mov qword ptr [ArithmeticProject!b (0000000000404008)],0x1
00000000`00401026 488b05d32f0000         mov rax,[ArithmeticProject!a (0000000000404000)]
00000000`0040102d 480105d42f0000         add [ArithmeticProject!b (0000000000404008)],rax
00000000`00401034 48ffc0                 inc     rax
00000000`00401037 48f72dca2f0000         imul qword ptr [ArithmeticProject!b (0000000000404008)]
00000000`0040103e 488905c32f0000         mov [ArithmeticProject!b (0000000000404008)],rax
00000000`00401045 c3                     ret

0:000> bp 00000000`00401010

0:000> g
Breakpoint 0 hit
ArithmeticProject!main:
00000000`00401010 48c705e52f000001000000 mov qword ptr [ArithmeticProject!a (0000000000404000)],0x1
      ds:00000000`00404000=0000000000000000

0:000> u
ArithmeticProject!main [C:\dmitri\training-64\Part1\Project1\ArithmeticProject.asm @ 13]:
00000000`00401010 48c705e52f000001000000 mov qword ptr [ArithmeticProject!a (0000000000404000)],0x1
00000000`0040101b 48c705e22f000001000000 mov qword ptr [ArithmeticProject!b (0000000000404008)],0x1
00000000`00401026 488b05d32f0000         mov rax,[ArithmeticProject!a (0000000000404000)]
00000000`0040102d 480105d42f0000         add [ArithmeticProject!b (0000000000404008)],rax
00000000`00401034 48ffc0                 inc     rax
00000000`00401037 48f72dca2f0000         imul qword ptr [ArithmeticProject!b (0000000000404008)]
00000000`0040103e 488905c32f0000         mov [ArithmeticProject!b (0000000000404008)],rax
00000000`00401045 c3                     ret
```

Picture 2


Address (Location) : 0
0
Location : a (Address 0000000000404000)
0
Location : b (Address 0000000000404008)
0
Address (Location) : 0000000000404010
0
0
0
Register RAX
0

Arithmetic Project - Calculations

[a] := 1

[b] := 1

[b] := [b] + [a] *; b = 2*

[b] := [b] * 2 *; b = 4*

Computer program

- We can think of a computer program as a sequence of instructions for manipulation of contents of memory cells and registers
- For example, addition: add the contents of memory cell №12 to the contents of memory cell №14.

  In notation [14] := [14] + [12]
- Because memory cells contain “numbers” we start with simple arithmetic

Assigning numbers to memory cells

- [a] := 1
- “a” means location (address) of the memory cell
  the name of location (address) 0000000000404000
  [a] means contents at the address “a”
- In C language “a” is called “variable” and we write:

```
a = 1;
```

- In assembly language we write:

```
mov [a], 1
```

- In WinDbg disassembly output we see:

mov qword ptr [ArithmeticProject!a (0000000000404000)],0x1

Arithmetic Project - Calculations

[a] := 1

[b] := 1

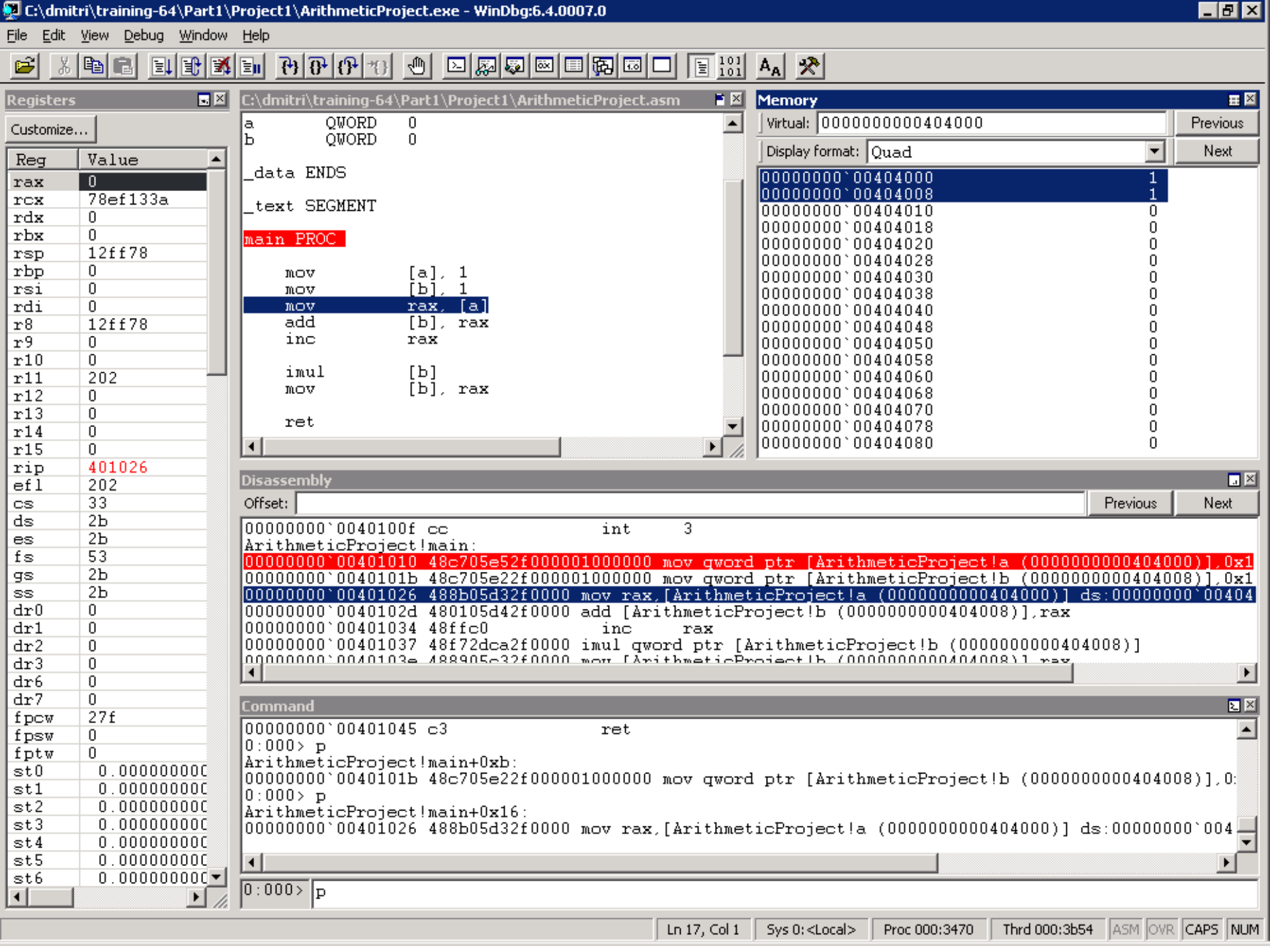
[b] := [b] + [a] *; b = 2*

[b] := [b] * 2 *; b = 4*

```
mov [a], 1
mov [b], 1
```

Picture 3


Address (Location) : 0
0
Location : a (Address 0000000000404000)
1
Location : b (Address 0000000000404008)
1
Address (Location) : 0000000000404010
0
0
0
Register RAX
0


C:\dmitri\training-64\Part1\Project1\ArithmeticProject.exe - WinDbg:6.4.0007.0
File Edit View Debug Window Help
Registers
Customize...
Reg Value
rax 0
rcx 78ef133a
rdx 0
rbx 0
rsp 12ff78
rbp 0
rsi 0
rdi 0
r8 12ff78
r9 0
r10 0
r11 202
r12 0
r13 0
r14 0
r15 0
rip 401026
efl 202
cs 33
ds 2b
es 2b
fs 53
gs 2b
ss 2b
dr0 0
dr1 0
dr2 0
dr3 0
dr6 0
dr7 0
fpcw 27f
fpsw 0
fptw 0
st0 0.000000000
st1 0.000000000
st2 0.000000000
st3 0.000000000
st4 0.000000000
st5 0.000000000
st6 0.000000000
C:\dmitri\training-64\Part1\Project1\ArithmeticProject.asm
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
imul [b]
mov [b], rax
ret
Memory
Virtual: 0000000000404000
Previous
Display format: Quad
Next
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
00000000`00404060 0
00000000`00404068 0
00000000`00404070 0
00000000`00404078 0
00000000`00404080 0
Disassembly
Offset:
Previous
Next
00000000`0040100f cc int 3
ArithmeticProject!main:
00000000`00401010 48c705e52f000001000000 mov qword ptr [ArithmeticProject!a (0000000000404000)],0x1
00000000`0040101b 48c705e22f000001000000 mov qword ptr [ArithmeticProject!b (0000000000404008)],0x1
00000000`00401026 488b05d32f0000 mov rax,[ArithmeticProject!a (0000000000404000)] ds:00000000`00404
00000000`0040102d 480105d42f0000 add [ArithmeticProject!b (0000000000404008)],rax
00000000`00401034 48ffc0 inc rax
00000000`00401037 48f72dca2f0000 imul qword ptr [ArithmeticProject!b (0000000000404008)]
Command
00000000`00401045 c3 ret
0:000> p
ArithmeticProject!main+0xb:
00000000`0040101b 48c705e22f000001000000 mov qword ptr [ArithmeticProject!b (0000000000404008)],0:
0:000> p
ArithmeticProject!main+0x16:
00000000`00401026 488b05d32f0000 mov rax,[ArithmeticProject!a (0000000000404000)] ds:00000000`004
0:000> p
Ln 17, Col 1
Sys 0:<Local>
Proc 000:3470
Thrd 000:3b54
ASM
OVR
CAPS
NUM

Assigning numbers to registers

- register := 1 or register := [a]
- We do not use brackets when refer to register contents
- Latter instruction means assign (copy) the number at the location (address) “a” to a register
- In assembly language we write:

```
mov rax, 1
mov rax, [a]
```

- In WinDbg disassembly output we see:

```
mov rax,[ArithmeticProject!a (0000000000404000)]
```

Adding numbers to memory cells

- [b] := [b] + [a]
- “a” and “b” mean locations (addresses) “a” and “b”, names of locations (addresses) 0000000000404000 and 0000000000404008. [a] and [b] mean contents at the adresses “a” and “b”, simply some numbers
- In C language we write:

```
b = b + a;
```

- In assembly language we use instruction **add**
- We cannot use both memory addresses in one step (instruction): **add [b], [a]**
  We can only use **add [b], register**

  Here **register** is like a temporary memory cell:

```
register := [a]
[b] := [b] + register
```

- In assembly language we write:

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

- In WinDbg disassembly output we see:

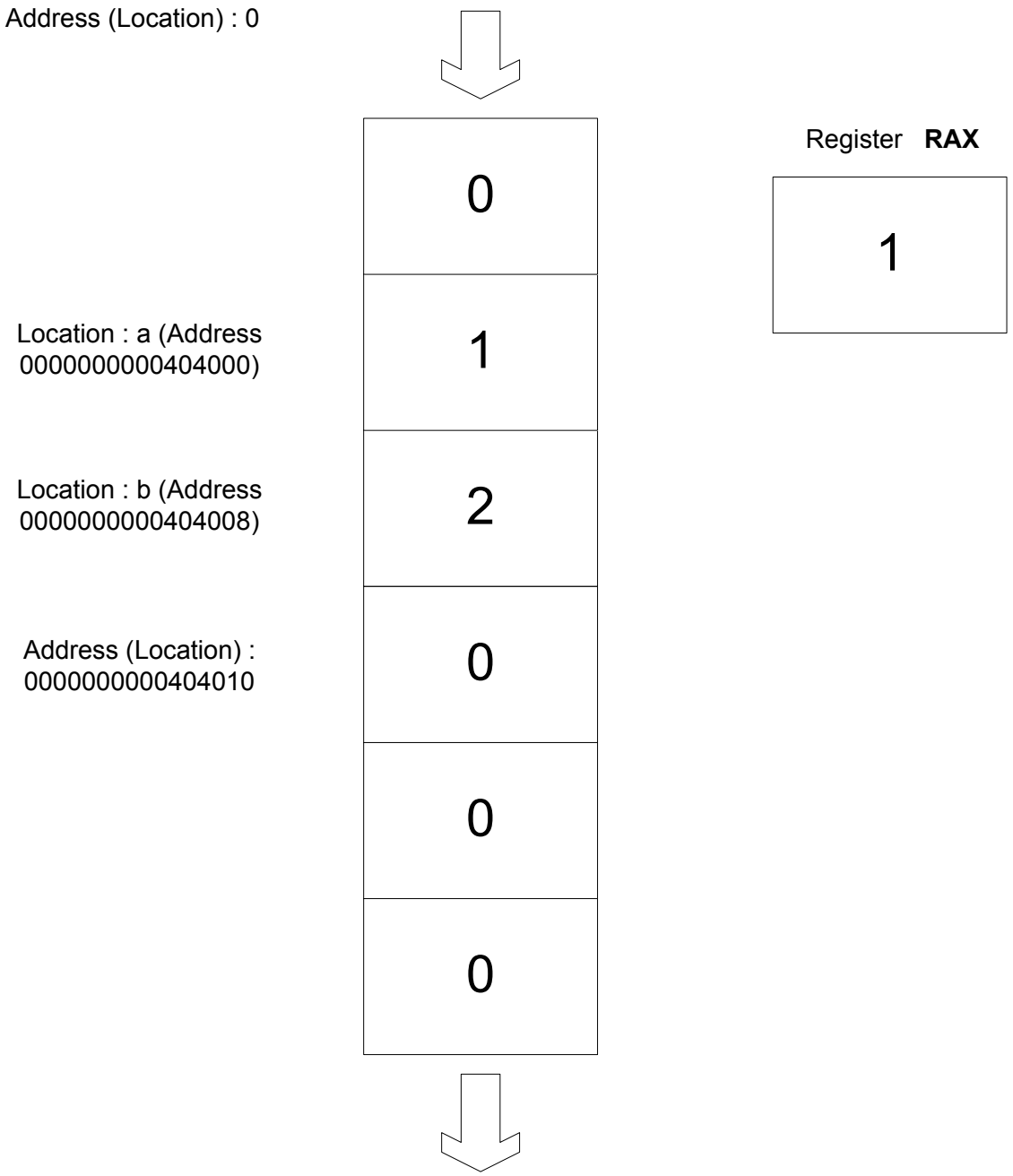
```
mov rax,[ArithmeticProject!a (0000000000404000)]
add [ArithmeticProject!b (0000000000404008)],rax
```

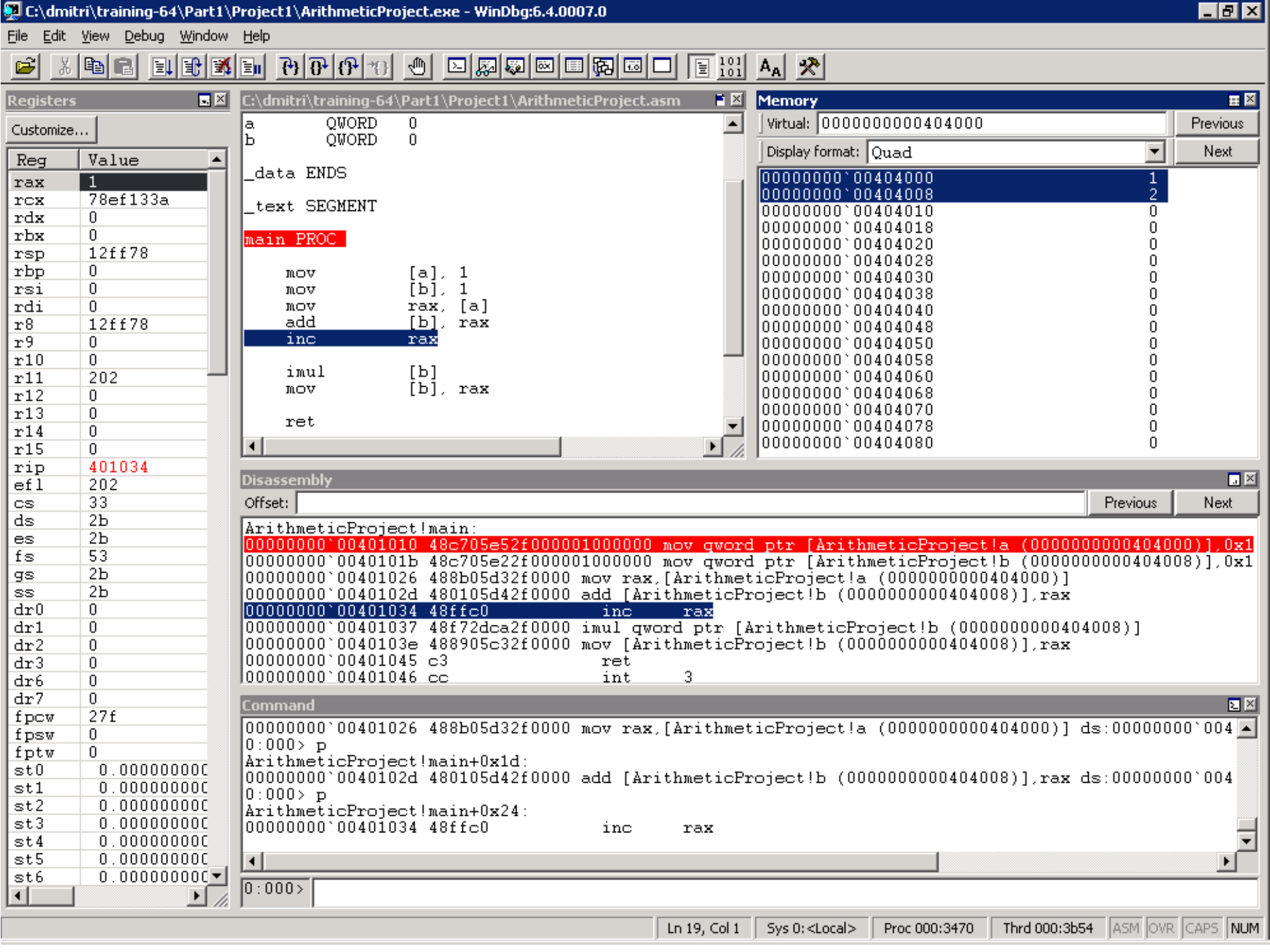
Arithmetic Project - Calculations

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

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

Picture 4


Address (Location) : 0
0
Location : a (Address 0000000000404000)
1
Location : b (Address 0000000000404008)
2
Address (Location) : 0000000000404010
0
0
0
Register RAX
1


C:\dmitri\training-64\Part1\Project1\ArithmeticProject.exe - WinDbg:6.4.0007.0
File Edit View Debug Window Help
Registers
Customize...
Reg Value
rax 1
rcx 78ef133a
rdx 0
rbx 0
rsp 12ff78
rbp 0
rsi 0
rdi 0
r8 12ff78
r9 0
r10 0
r11 202
r12 0
r13 0
r14 0
r15 0
rip 401034
efl 202
cs 33
ds 2b
es 2b
fs 53
gs 2b
ss 2b
dr0 0
dr1 0
dr2 0
dr3 0
dr6 0
dr7 0
fpcw 27f
fpsw 0
fptw 0
st0 0.000000000
st1 0.000000000
st2 0.000000000
st3 0.000000000
st4 0.000000000
st5 0.000000000
st6 0.000000000
C:\dmitri\training-64\Part1\Project1\ArithmeticProject.asm
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
imul [b]
mov [b], rax
ret
Memory
Virtual: 0000000000404000
Previous
Display format: Quad
Next
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
00000000`00404060 0
00000000`00404068 0
00000000`00404070 0
00000000`00404078 0
00000000`00404080 0
Disassembly
Offset:
Previous
Next
ArithmeticProject!main:
00000000`00401010 48c705e52f000001000000 mov qword ptr [ArithmeticProject!a (0000000000404000)],0x1
00000000`0040101b 48c705e22f000001000000 mov qword ptr [ArithmeticProject!b (0000000000404008)],0x1
00000000`00401026 488b05d32f0000 mov rax,[ArithmeticProject!a (0000000000404000)]
00000000`0040102d 480105d42f0000 add [ArithmeticProject!b (0000000000404008)],rax
00000000`00401034 48ffc0 inc rax
00000000`00401037 48f72dca2f0000 imul qword ptr [ArithmeticProject!b (0000000000404008)]
00000000`0040103e 488905c32f0000 mov [ArithmeticProject!b (0000000000404008)],rax
00000000`00401045 c3 ret
00000000`00401046 cc int 3
Command
00000000`00401026 488b05d32f0000 mov rax,[ArithmeticProject!a (0000000000404000)] ds:00000000`004
0:000> p
ArithmeticProject!main+0x1d:
00000000`0040102d 480105d42f0000 add [ArithmeticProject!b (0000000000404008)],rax ds:00000000`004
0:000> p
ArithmeticProject!main+0x24:
00000000`00401034 48ffc0 inc rax
0:000>
Ln 19, Col 1
Sys 0:<Local>
Proc 000:3470
Thrd 000:3b54
ASM
OVR
CAPS
NUM

Incrementing (Decrementing) numbers in memory cells and registers

- [a] := [a] + 1 ([a] := [a] – 1)

  Means increment (decrement) number at location (address) “a”

- In C language we write:

```
a = a + 1; or ++a; or a++;
b = b – 1; or --b; or b--;
```

- In assembly language we use instructions **inc** and **dec**

- In assembly language we write:

```
inc   [a]
inc   rax
dec   [a]
dec   rax
```

- In WinDbg disassembly output we see:

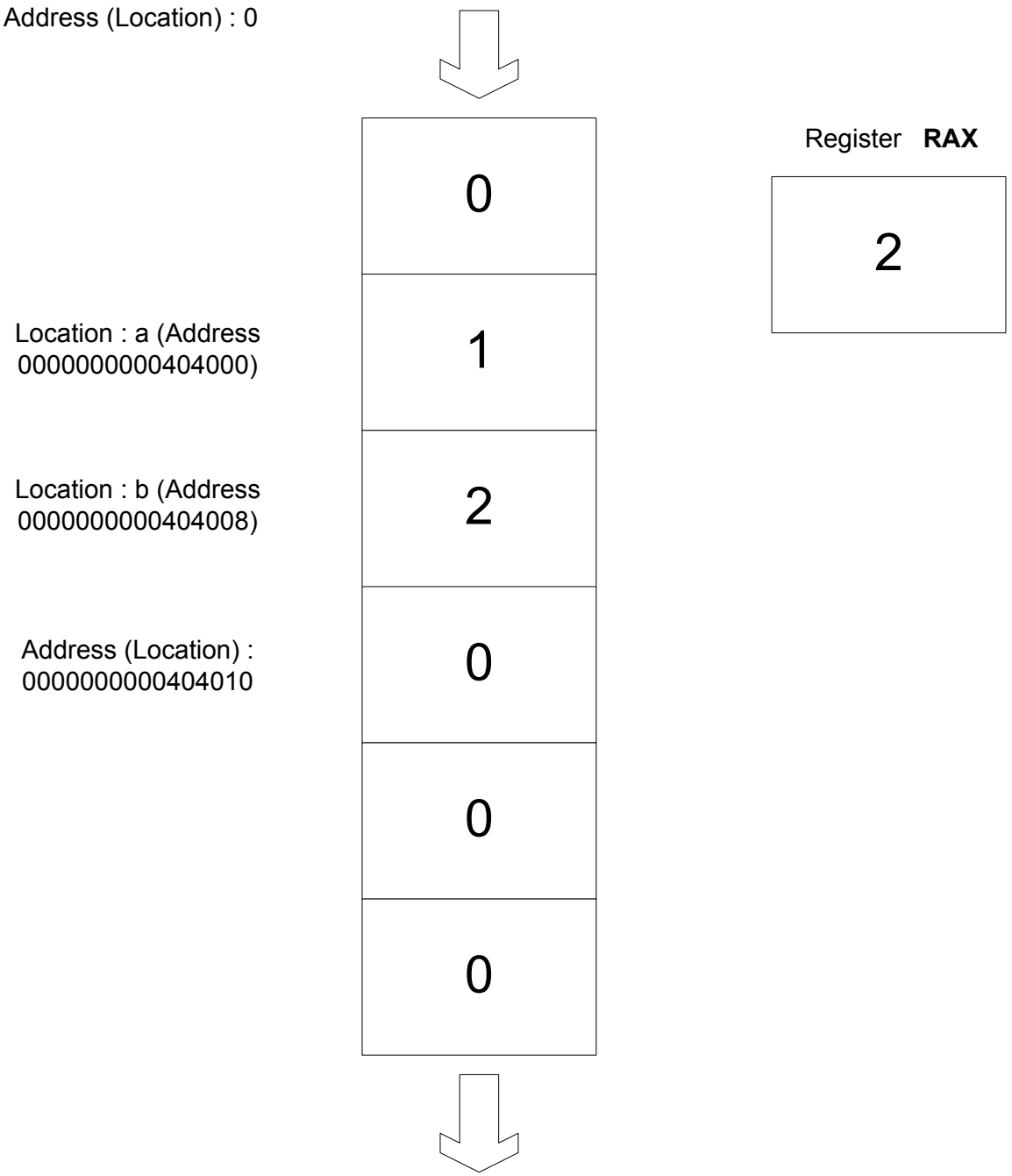
```
inc  rax
```

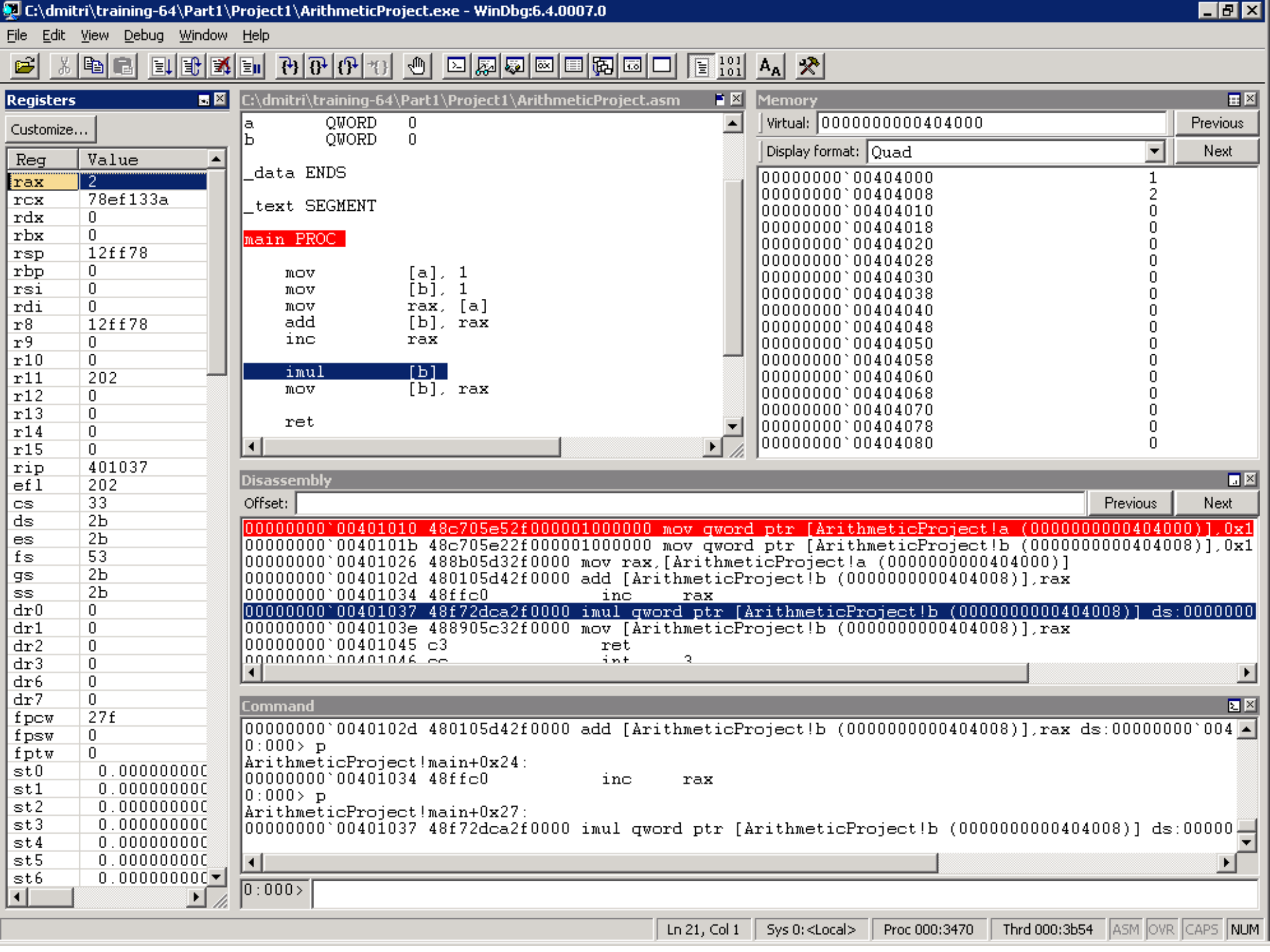
Arithmetic Project - Calculations

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

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

Picture 5


Address (Location) : 0
0
Location : a (Address 0000000000404000)
1
Location : b (Address 0000000000404008)
2
Address (Location) : 0000000000404010
0
0
0
Register RAX
2


C:\dmitri\training-64\Part1\Project1\ArithmeticProject.exe - WinDbg:6.4.0007.0
File Edit View Debug Window Help
Registers
Customize...
Reg Value
rax 2
rcx 78ef133a
rdx 0
rbx 0
rsp 12ff78
rbp 0
rsi 0
rdi 0
r8 12ff78
r9 0
r10 0
r11 202
r12 0
r13 0
r14 0
r15 0
rip 401037
efl 202
cs 33
ds 2b
es 2b
fs 53
gs 2b
ss 2b
dr0 0
dr1 0
dr2 0
dr3 0
dr6 0
dr7 0
fpcw 27f
fpsw 0
fptw 0
st0 0.000000000
st1 0.000000000
st2 0.000000000
st3 0.000000000
st4 0.000000000
st5 0.000000000
st6 0.000000000
C:\dmitri\training-64\Part1\Project1\ArithmeticProject.asm
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
imul [b]
mov [b], rax
ret
Memory
Virtual: 0000000000404000
Previous
Display format: Quad
Next
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
00000000`00404060 0
00000000`00404068 0
00000000`00404070 0
00000000`00404078 0
00000000`00404080 0
Disassembly
Offset:
Previous
Next
00000000`00401010 48c705e52f000001000000 mov qword ptr [ArithmeticProject!a (0000000000404000)],0x1
00000000`0040101b 48c705e22f000001000000 mov qword ptr [ArithmeticProject!b (0000000000404008)],0x1
00000000`00401026 488b05d32f0000 mov rax,[ArithmeticProject!a (0000000000404000)]
00000000`0040102d 480105d42f0000 add [ArithmeticProject!b (0000000000404008)],rax
00000000`00401034 48ffc0 inc rax
00000000`00401037 48f72dca2f0000 imul qword ptr [ArithmeticProject!b (0000000000404008)] ds:0000000
00000000`0040103e 488905c32f0000 mov [ArithmeticProject!b (0000000000404008)],rax
00000000`00401045 c3 ret
Command
00000000`0040102d 480105d42f0000 add [ArithmeticProject!b (0000000000404008)],rax ds:00000000`004
0:000> p
ArithmeticProject!main+0x24:
00000000`00401034 48ffc0 inc rax
0:000> p
ArithmeticProject!main+0x27:
00000000`00401037 48f72dca2f0000 imul qword ptr [ArithmeticProject!b (0000000000404008)] ds:00000
0:000>
Ln 21, Col 1
Sys 0:<Local>
Proc 000:3470
Thrd 000:3b54
ASM
OVR
CAPS
NUM

Multiplying numbers

- [b] := [b] * 2

  Means multiply the number at the location (address) “b” by 2

- In C language we write:

```
b =  b * 2;
b *= 2;
```

- In assembly language we use instruction **imul** (integer multiply)

- In assembly language we write:

```
imul [b]
mov  [b], rax
```

  Means [b] := [b] * rax, so we have to put 2 into rax, but we already have 2 in rax
  Result of multiplication is put into registers rax and rdx

- In WinDbg disassembly output we see:

```
imul qword ptr [ArithmeticProject!b (0000000000404008)]
mov [ArithmeticProject!b (0000000000404008)],rax
```

Multiplication and registers

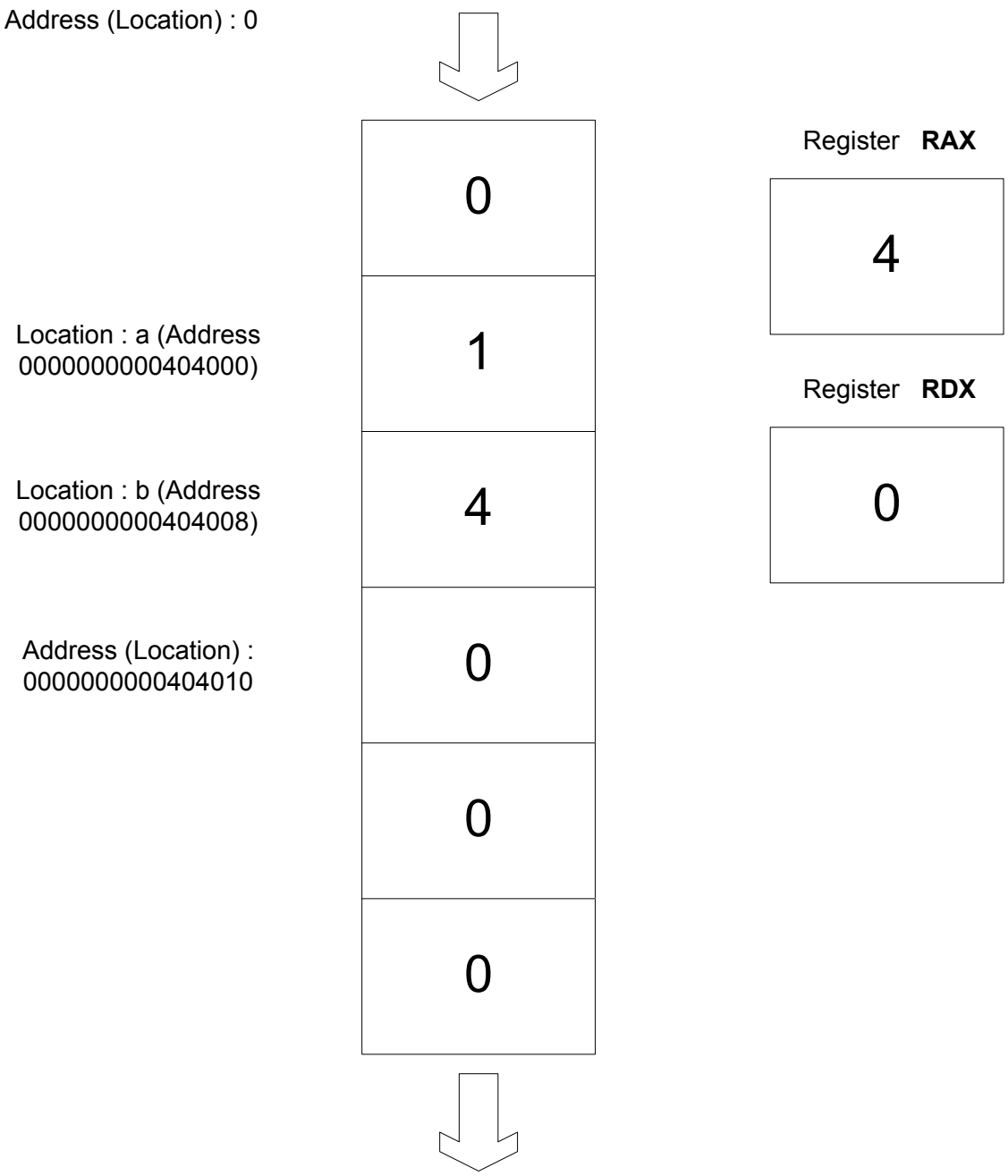
- Why the result of multiplication occupies two registers rax and rdx?
- Each 64bit register or memory cell can contain any number between -9223372036854775808 and 9223372036854775807
- If we multiply 2 by 2 the result can be put into one register rax
- If we multiply 9223372036854775807 by 4 we get 36893488147419103228. The result is too big to fit into one register or memory cell.
- We can think of rdx:rax pair as two memory cells joined together to hold the multiplication result.

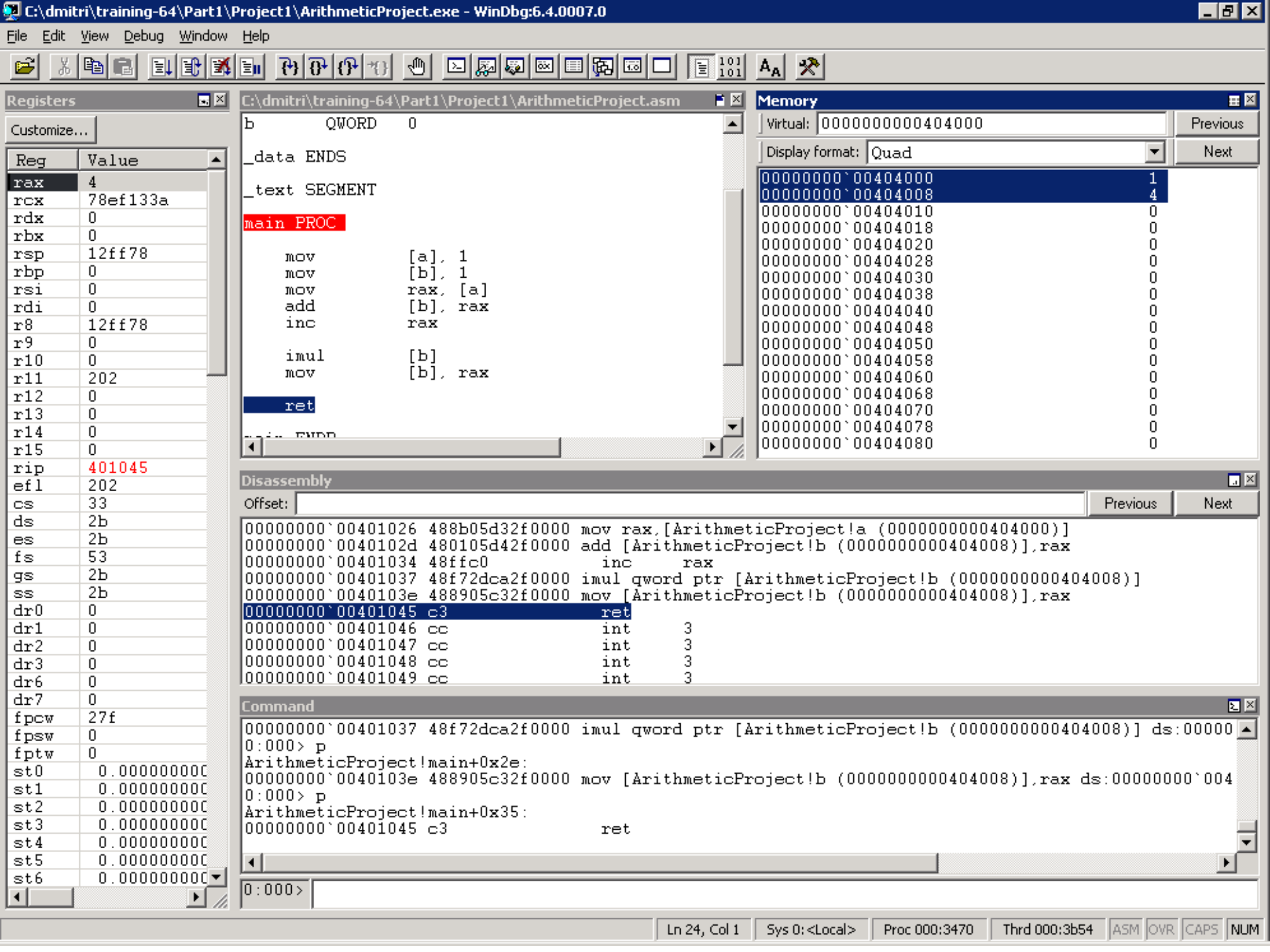
Arithmetic Project - Calculations

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

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

imul [b]
mov [b], rax
```

Picture 6


Address (Location) : 0
0
Location : a (Address 0000000000404000)
1
Location : b (Address 0000000000404008)
4
Address (Location) : 0000000000404010
0
0
0
Register RAX
4
Register RDX
0


C:\dmitri\training-64\Part1\Project1\ArithmeticProject.exe - WinDbg:6.4.0007.0
File Edit View Debug Window Help
Registers
Customize...
Reg Value
rax 4
rcx 78ef133a
rdx 0
rbx 0
rsp 12ff78
rbp 0
rsi 0
rdi 0
r8 12ff78
r9 0
r10 0
r11 202
r12 0
r13 0
r14 0
r15 0
rip 401045
efl 202
cs 33
ds 2b
es 2b
fs 53
gs 2b
ss 2b
dr0 0
dr1 0
dr2 0
dr3 0
dr6 0
dr7 0
fpcw 27f
fpsw 0
fptw 0
st0 0.000000000
st1 0.000000000
st2 0.000000000
st3 0.000000000
st4 0.000000000
st5 0.000000000
st6 0.000000000
C:\dmitri\training-64\Part1\Project1\ArithmeticProject.asm
b QWORD 0
_data ENDS
_text SEGMENT
main PROC
mov [a], 1
mov [b], 1
mov rax, [a]
add [b], rax
inc rax
imul [b]
mov [b], rax
ret
Memory
Virtual: 0000000000404000
Previous
Display format: Quad
Next
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
00000000`00404060 0
00000000`00404068 0
00000000`00404070 0
00000000`00404078 0
00000000`00404080 0
Disassembly
Offset:
Previous
Next
00000000`00401026 488b05d32f0000 mov rax,[ArithmeticProject!a (0000000000404000)]
00000000`0040102d 480105d42f0000 add [ArithmeticProject!b (0000000000404008)],rax
00000000`00401034 48ffc0 inc rax
00000000`00401037 48f72dca2f0000 imul qword ptr [ArithmeticProject!b (0000000000404008)]
00000000`0040103e 488905c32f0000 mov [ArithmeticProject!b (0000000000404008)],rax
00000000`00401045 c3 ret
00000000`00401046 cc int 3
00000000`00401047 cc int 3
00000000`00401048 cc int 3
00000000`00401049 cc int 3
Command
00000000`00401037 48f72dca2f0000 imul qword ptr [ArithmeticProject!b (0000000000404008)] ds:00000
0:000> p
ArithmeticProject!main+0x2e:
00000000`0040103e 488905c32f0000 mov [ArithmeticProject!b (0000000000404008)],rax ds:00000000`004
0:000> p
ArithmeticProject!main+0x35:
00000000`00401045 c3 ret
0:000>
Ln 24, Col 1
Sys 0:<Local>
Proc 000:3470
Thrd 000:3b54
ASM
OVR
CAPS
NUM

What's next?

- We will review today's material by looking at the x64 disassembly output of the very simple program written in C.
- Representation of numbers. Hexadecimal notation.
- The concept of the "pointer".
- We will rewrite our arithmetic project using pointers.