Just Enough Assembly for Compiler Explorer

Anders Schau Knatten

MUREX

CODE RECKONS

Science to the CORE


KUKA

C++ Quiz

You've answered 1 of 138 questions correctly. (Clear)

Question #197 Difficulty: ●○○

According to the C++17 standard, what is the output of this program?

```
#include <iostream>

int j = 1;

int main() {
  int& i = j, j;
  j = 2;
  std::cout << i << j;
}
```

Answer: The program is guaranteed to output: Answer

Problems? View a hint or try another question.

I give up, show me the answer (make 3 more attempts first).

Mode : Training

You are currently in training mode, answering random questions. Why not Start a new quiz? Then you can boast about your score, and invite your friends.

Contribute

Create your own!

Android app

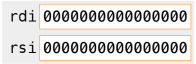
Get Sergey Vasilchenko's CppQuiz Android app.

@knatten / @CppQuiz / @AffectiveCpp

Compiler Explorer

https://godbolt.org/z/qTeo75jrr

Registers



Registers

```
mov rdi, 2
mov rsi, 4
add rdi, rsi
```

Registers

```
mov rdi, 2
mov rsi, 4
add rdi, rsi
```

Registers

```
mov rdi, 2
mov rsi, 4
add rdi, rsi
```

Registers

```
mov rdi, 2
mov rsi, 4
add rdi, rsi
```

Stack push/pop

```
mov  rdi, 0xfedcba9876543210
push rdi
pop  rsi
```

rdi	0000000000000000
rsi	0000000000000000
rsp	00000000000000b0

	…00b3	00
	…00b2	00
	…00b1	00
rsp→	…00b0	00
	…00af	00
	…00ae	00
	…00ad	00
	…00ac	00
	…00ab	00
	…00aa	00
	…00a9	00
	…00a8	00
	…00a7	00
	…00a6	00
	…00a5	00
	…00a4	00
	…00a3	00
	…00a2	00
	…00a1	00
	…00a0	00

Stack push/pop

```
mov  rdi, 0xfedcba9876543210
push rdi
pop  rsi
```

rdi	fedcba9876543210
rsi	0000000000000000
rsp	00000000000000b0

	…00b3	00
	…00b2	00
	…00b1	00
rsp→	…00b0	00
	…00af	00
	…00ae	00
	…00ad	00
	…00ac	00
	…00ab	00
	…00aa	00
	…00a9	00
	…00a8	00
	…00a7	00
	…00a6	00
	…00a5	00
	…00a4	00
	…00a3	00
	…00a2	00
	…00a1	00
	…00a0	00

Stack push/pop

```
mov  rdi, 0xfedcba9876543210
push rdi
pop  rsi
```

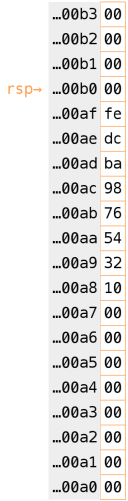
rdi	fedcba9876543210
rsi	0000000000000000
rsp	00000000000000a8

	…00b3	00
	…00b2	00
	…00b1	00
	…00b0	00
	…00af	fe
	…00ae	dc
	…00ad	ba
	…00ac	98
	…00ab	76
	…00aa	54
	…00a9	32
rsp→	…00a8	10
	…00a7	00
	…00a6	00
	…00a5	00
	…00a4	00
	…00a3	00
	…00a2	00
	…00a1	00
	…00a0	00

Stack push/pop

```
mov  rdi, 0xfedcba9876543210
push rdi
pop  rsi
```

rdi	fedcba9876543210
rsi	fedcba9876543210
rsp	00000000000000b0



Addressing memory

```
mov rsi, 0x3333333333333333
mov rdi, 0x2222222222222222
mov qword ptr [rsp - 8], rsi
add rdi, qword ptr [rsp - 8]
```

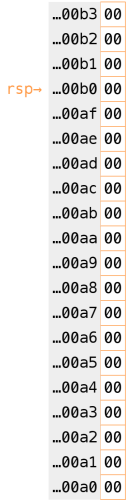
rsi	0000000000000000
rdi	0000000000000000
rsp	00000000000000b0

	…00b3	00
	…00b2	00
	…00b1	00
rsp→	…00b0	00
	…00af	00
	…00ae	00
	…00ad	00
	…00ac	00
	…00ab	00
	…00aa	00
	…00a9	00
	…00a8	00
	…00a7	00
	…00a6	00
	…00a5	00
	…00a4	00
	…00a3	00
	…00a2	00
	…00a1	00
	…00a0	00

Addressing memory

```
mov rsi, 0x3333333333333333
mov rdi, 0x2222222222222222
mov qword ptr [rsp - 8], rsi
add rdi, qword ptr [rsp - 8]
```

rsi	3333333333333333
rdi	0000000000000000
rsp	00000000000000b0



Addressing memory

```
mov rsi, 0x3333333333333333
mov rdi, 0x2222222222222222
mov qword ptr [rsp - 8], rsi
add rdi, qword ptr [rsp - 8]
```

rsi	3333333333333333
rdi	2222222222222222
rsp	00000000000000b0

	…00b3	00
	…00b2	00
	…00b1	00
rsp→	…00b0	00
	…00af	00
	…00ae	00
	…00ad	00
	…00ac	00
	…00ab	00
	…00aa	00
	…00a9	00
	…00a8	00
	…00a7	00
	…00a6	00
	…00a5	00
	…00a4	00
	…00a3	00
	…00a2	00
	…00a1	00
	…00a0	00

Addressing memory

```
mov rsi, 0x3333333333333333
mov rdi, 0x2222222222222222
mov qword ptr [rsp - 8], rsi
add rdi, qword ptr [rsp - 8]
```

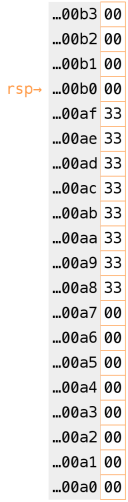
rsi	3333333333333333
rdi	2222222222222222
rsp	00000000000000b0

	…00b3	00
	…00b2	00
	…00b1	00
rsp→	…00b0	00
	…00af	33
	…00ae	33
	…00ad	33
	…00ac	33
	…00ab	33
	…00aa	33
	…00a9	33
	…00a8	33
	…00a7	00
	…00a6	00
	…00a5	00
	…00a4	00
	…00a3	00
	…00a2	00
	…00a1	00
	…00a0	00

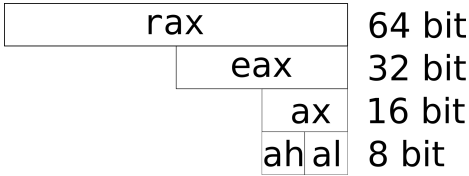
Addressing memory

```
mov rsi, 0x3333333333333333
mov rdi, 0x2222222222222222
mov qword ptr [rsp - 8], rsi
add rdi, qword ptr [rsp - 8]
```

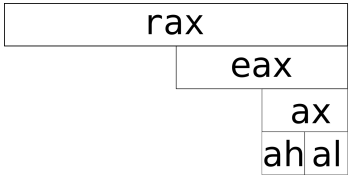
rsi	3333333333333333
rdi	5555555555555555
rsp	00000000000000b0



Register sizes



Register sizes



64 bit
32 bit
16 bit
8 bit

- (signed / unsigned) long
- (signed / unsigned) int
- (signed / unsigned) short
- (signed / unsigned) char

32 bit register

```
mov esi, 0x33333333
mov edi, 0x22222222
mov dword ptr [rsp - 4], esi
add edi, dword ptr [rsp - 4]
```

rsi	0000000000000000
rdi	0000000000000000
rsp	00000000000000b0

	…00b3	00
	…00b2	00
	…00b1	00
rsp→	…00b0	00
	…00af	00
	…00ae	00
	…00ad	00
	…00ac	00
	…00ab	00
	…00aa	00
	…00a9	00
	…00a8	00
	…00a7	00
	…00a6	00
	…00a5	00
	…00a4	00
	…00a3	00
	…00a2	00
	…00a1	00
	…00a0	00

32 bit register

```
mov esi, 0x33333333
mov edi, 0x22222222
mov dword ptr [rsp - 4], esi
add edi, dword ptr [rsp - 4]
```

rsi	0000000033333333
rdi	0000000000000000
rsp	00000000000000b0

	…00b3	00
	…00b2	00
	…00b1	00
rsp→	…00b0	00
	…00af	00
	…00ae	00
	…00ad	00
	…00ac	00
	…00ab	00
	…00aa	00
	…00a9	00
	…00a8	00
	…00a7	00
	…00a6	00
	…00a5	00
	…00a4	00
	…00a3	00
	…00a2	00
	…00a1	00
	…00a0	00

32 bit register

```
mov esi, 0x33333333
mov edi, 0x22222222
mov dword ptr [rsp - 4], esi
add edi, dword ptr [rsp - 4]
```

rsi	0000000033333333
rdi	0000000022222222
rsp	00000000000000b0

	…00b3	00
	…00b2	00
	…00b1	00
rsp→	…00b0	00
	…00af	00
	…00ae	00
	…00ad	00
	…00ac	00
	…00ab	00
	…00aa	00
	…00a9	00
	…00a8	00
	…00a7	00
	…00a6	00
	…00a5	00
	…00a4	00
	…00a3	00
	…00a2	00
	…00a1	00
	…00a0	00

32 bit register

```
mov esi, 0x33333333
mov edi, 0x22222222
mov dword ptr [rsp - 4], esi
add edi, dword ptr [rsp - 4]
```

rsi	0000000033333333
rdi	0000000022222222
rsp	00000000000000b0

	…00b3	00
	…00b2	00
	…00b1	00
rsp→	…00b0	00
	…00af	33
	…00ae	33
	…00ad	33
	…00ac	33
	…00ab	00
	…00aa	00
	…00a9	00
	…00a8	00
	…00a7	00
	…00a6	00
	…00a5	00
	…00a4	00
	…00a3	00
	…00a2	00
	…00a1	00
	…00a0	00

32 bit register

```
mov esi, 0x33333333
mov edi, 0x22222222
mov dword ptr [rsp - 4], esi
add edi, dword ptr [rsp - 4]
```

rsi	0000000033333333
rdi	0000000055555555
rsp	00000000000000b0

	…00b3	00
	…00b2	00
	…00b1	00
rsp→	…00b0	00
	…00af	33
	…00ae	33
	…00ad	33
	…00ac	33
	…00ab	00
	…00aa	00
	…00a9	00
	…00a8	00
	…00a7	00
	…00a6	00
	…00a5	00
	…00a4	00
	…00a3	00
	…00a2	00
	…00a1	00
	…00a0	00

More addressing

```
mov word ptr [rsp-8], 0x1111
mov word ptr [rsp-6], 0x2222
mov word ptr [rsp-4], 0x3333
mov word ptr [rsp-2], 0x4444
lea rdi, [rsp - 8]
mov ax, word ptr [rdi + 6]
mov rsi, 0
mov ax, word ptr [rdi + rsi * 2]
inc rsi
mov ax, word ptr [rdi + rsi * 2]
```

rsi	0000000000000000
rdi	0000000000000000
rax	0000000000000000
rsp	00000000000000b0

	…00b3	00
	…00b2	00
	…00b1	00
rsp→	…00b0	00
	…00af	00
	…00ae	00
	…00ad	00
	…00ac	00
	…00ab	00
	…00aa	00
	…00a9	00
	…00a8	00
	…00a7	00
	…00a6	00
	…00a5	00
	…00a4	00
	…00a3	00
	…00a2	00
	…00a1	00
	…00a0	00

More addressing

```
mov word ptr [rsp-8], 0x1111
mov word ptr [rsp-6], 0x2222
mov word ptr [rsp-4], 0x3333
mov word ptr [rsp-2], 0x4444
lea rdi, [rsp - 8]
mov ax, word ptr [rdi + 6]
mov rsi, 0
mov ax, word ptr [rdi + rsi * 2]
inc rsi
mov ax, word ptr [rdi + rsi * 2]
```

rsi	0000000000000000
rdi	0000000000000000
rax	0000000000000000
rsp	00000000000000b0

	…00b3	00
	…00b2	00
	…00b1	00
rsp→	…00b0	00
	…00af	00
	…00ae	00
	…00ad	00
	…00ac	00
	…00ab	00
	…00aa	00
	…00a9	11
	…00a8	11
	…00a7	00
	…00a6	00
	…00a5	00
	…00a4	00
	…00a3	00
	…00a2	00
	…00a1	00
	…00a0	00

More addressing

```
mov word ptr [rsp-8], 0x1111
mov word ptr [rsp-6], 0x2222
mov word ptr [rsp-4], 0x3333
mov word ptr [rsp-2], 0x4444
lea rdi, [rsp - 8]
mov ax, word ptr [rdi + 6]
mov rsi, 0
mov ax, word ptr [rdi + rsi * 2]
inc rsi
mov ax, word ptr [rdi + rsi * 2]
```

rsi	0000000000000000
rdi	0000000000000000
rax	0000000000000000
rsp	00000000000000b0

	…00b3	00
	…00b2	00
	…00b1	00
rsp→	…00b0	00
	…00af	00
	…00ae	00
	…00ad	00
	…00ac	00
	…00ab	22
	…00aa	22
	…00a9	11
	…00a8	11
	…00a7	00
	…00a6	00
	…00a5	00
	…00a4	00
	…00a3	00
	…00a2	00
	…00a1	00
	…00a0	00

More addressing

```
mov word ptr [rsp-8], 0x1111
mov word ptr [rsp-6], 0x2222
mov word ptr [rsp-4], 0x3333
mov word ptr [rsp-2], 0x4444
lea rdi, [rsp - 8]
mov ax, word ptr [rdi + 6]
mov rsi, 0
mov ax, word ptr [rdi + rsi * 2]
inc rsi
mov ax, word ptr [rdi + rsi * 2]
```

rsi	0000000000000000
rdi	0000000000000000
rax	0000000000000000
rsp	00000000000000b0

	…00b3	00
	…00b2	00
	…00b1	00
rsp→	…00b0	00
	…00af	00
	…00ae	00
	…00ad	33
	…00ac	33
	…00ab	22
	…00aa	22
	…00a9	11
	…00a8	11
	…00a7	00
	…00a6	00
	…00a5	00
	…00a4	00
	…00a3	00
	…00a2	00
	…00a1	00
	…00a0	00

More addressing

```
mov word ptr [rsp-8], 0x1111
mov word ptr [rsp-6], 0x2222
mov word ptr [rsp-4], 0x3333
mov word ptr [rsp-2], 0x4444
lea rdi, [rsp - 8]
mov ax, word ptr [rdi + 6]
mov rsi, 0
mov ax, word ptr [rdi + rsi * 2]
inc rsi
mov ax, word ptr [rdi + rsi * 2]
```

rsi	0000000000000000
rdi	0000000000000000
rax	0000000000000000
rsp	00000000000000b0

	…00b3	00
	…00b2	00
	…00b1	00
rsp→	…00b0	00
	…00af	44
	…00ae	44
	…00ad	33
	…00ac	33
	…00ab	22
	…00aa	22
	…00a9	11
	…00a8	11
	…00a7	00
	…00a6	00
	…00a5	00
	…00a4	00
	…00a3	00
	…00a2	00
	…00a1	00
	…00a0	00

More addressing

```
mov word ptr [rsp-8], 0x1111
mov word ptr [rsp-6], 0x2222
mov word ptr [rsp-4], 0x3333
mov word ptr [rsp-2], 0x4444
lea rdi, [rsp - 8]
mov ax, word ptr [rdi + 6]
mov rsi, 0
mov ax, word ptr [rdi + rsi * 2]
inc rsi
mov ax, word ptr [rdi + rsi * 2]
```

rsi	0000000000000000
rdi	00000000000000a8
rax	0000000000000000
rsp	00000000000000b0

	…00b3	00
	…00b2	00
	…00b1	00
rsp→	…00b0	00
	…00af	44
	…00ae	44
	…00ad	33
	…00ac	33
	…00ab	22
	…00aa	22
	…00a9	11
	…00a8	11
	…00a7	00
	…00a6	00
	…00a5	00
	…00a4	00
	…00a3	00
	…00a2	00
	…00a1	00
	…00a0	00

More addressing

```
mov word ptr [rsp-8], 0x1111
mov word ptr [rsp-6], 0x2222
mov word ptr [rsp-4], 0x3333
mov word ptr [rsp-2], 0x4444
lea rdi, [rsp - 8]
mov ax, word ptr [rdi + 6]
mov rsi, 0
mov ax, word ptr [rdi + rsi * 2]
inc rsi
mov ax, word ptr [rdi + rsi * 2]
```

rsi	0000000000000000
rdi	00000000000000a8
rax	0000000000004444
rsp	00000000000000b0

	…00b3	00
	…00b2	00
	…00b1	00
rsp→	…00b0	00
	…00af	44
	…00ae	44
	…00ad	33
	…00ac	33
	…00ab	22
	…00aa	22
	…00a9	11
	…00a8	11
	…00a7	00
	…00a6	00
	…00a5	00
	…00a4	00
	…00a3	00
	…00a2	00
	…00a1	00
	…00a0	00

More addressing

```
mov word ptr [rsp-8], 0x1111
mov word ptr [rsp-6], 0x2222
mov word ptr [rsp-4], 0x3333
mov word ptr [rsp-2], 0x4444
lea rdi, [rsp - 8]
mov ax, word ptr [rdi + 6]
mov rsi, 0
mov ax, word ptr [rdi + rsi * 2]
inc rsi
mov ax, word ptr [rdi + rsi * 2]
```

rsi	0000000000000000
rdi	00000000000000a8
rax	0000000000004444
rsp	00000000000000b0

	…00b3	00
	…00b2	00
	…00b1	00
rsp→	…00b0	00
	…00af	44
	…00ae	44
	…00ad	33
	…00ac	33
	…00ab	22
	…00aa	22
	…00a9	11
	…00a8	11
	…00a7	00
	…00a6	00
	…00a5	00
	…00a4	00
	…00a3	00
	…00a2	00
	…00a1	00
	…00a0	00

More addressing

```
mov word ptr [rsp-8], 0x1111
mov word ptr [rsp-6], 0x2222
mov word ptr [rsp-4], 0x3333
mov word ptr [rsp-2], 0x4444
lea rdi, [rsp - 8]
mov ax, word ptr [rdi + 6]
mov rsi, 0
mov ax, word ptr [rdi + rsi * 2]
inc rsi
mov ax, word ptr [rdi + rsi * 2]
```

rsi	0000000000000000
rdi	00000000000000a8
rax	0000000000001111
rsp	00000000000000b0

	…00b3	00
	…00b2	00
	…00b1	00
rsp→	…00b0	00
	…00af	44
	…00ae	44
	…00ad	33
	…00ac	33
	…00ab	22
	…00aa	22
	…00a9	11
	…00a8	11
	…00a7	00
	…00a6	00
	…00a5	00
	…00a4	00
	…00a3	00
	…00a2	00
	…00a1	00
	…00a0	00

More addressing

```
mov word ptr [rsp-8], 0x1111
mov word ptr [rsp-6], 0x2222
mov word ptr [rsp-4], 0x3333
mov word ptr [rsp-2], 0x4444
lea rdi, [rsp - 8]
mov ax, word ptr [rdi + 6]
mov rsi, 0
mov ax, word ptr [rdi + rsi * 2]
inc rsi
mov ax, word ptr [rdi + rsi * 2]
```

rsi	0000000000000001
rdi	00000000000000a8
rax	0000000000001111
rsp	00000000000000b0

	…00b3	00
	…00b2	00
	…00b1	00
rsp→	…00b0	00
	…00af	44
	…00ae	44
	…00ad	33
	…00ac	33
	…00ab	22
	…00aa	22
	…00a9	11
	…00a8	11
	…00a7	00
	…00a6	00
	…00a5	00
	…00a4	00
	…00a3	00
	…00a2	00
	…00a1	00
	…00a0	00

More addressing

```
mov word ptr [rsp-8], 0x1111
mov word ptr [rsp-6], 0x2222
mov word ptr [rsp-4], 0x3333
mov word ptr [rsp-2], 0x4444
lea rdi, [rsp - 8]
mov ax, word ptr [rdi + 6]
mov rsi, 0
mov ax, word ptr [rdi + rsi * 2]
inc rsi
mov ax, word ptr [rdi + rsi * 2]
```

rsi	0000000000000001
rdi	00000000000000a8
rax	0000000000002222
rsp	00000000000000b0

	…00b3	00
	…00b2	00
	…00b1	00
rsp→	…00b0	00
	…00af	44
	…00ae	44
	…00ad	33
	…00ac	33
	…00ab	22
	…00aa	22
	…00a9	11
	…00a8	11
	…00a7	00
	…00a6	00
	…00a5	00
	…00a4	00
	…00a3	00
	…00a2	00
	…00a1	00
	…00a0	00

Conditionals

```
  mov  rdi, 3
  mov  rsi, 2
  cmp  rdi, rsi
  jg   .greater
  mov  rax, 0
  jmp .endif
.greater:
  mov rax, 1
.endif:
```

Conditionals

```
  mov  rdi, 3
  mov  rsi, 2
  cmp  rdi, rsi
  jg   .greater
  mov  rax, 0
  jmp .endif
.greater:
  mov rax, 1
.endif:
```

Conditionals

```
  mov  rdi, 3
  mov  rsi, 2
  cmp  rdi, rsi
  jg   .greater
  mov  rax, 0
  jmp .endif
.greater:
  mov rax, 1
.endif:
```

Conditionals

```
  mov  rdi, 3
  mov  rsi, 2
  cmp  rdi, rsi
  jg   .greater
  mov  rax, 0
  jmp .endif
.greater:
  mov rax, 1
.endif:
```

Conditionals

```
  mov  rdi, 3
  mov  rsi, 2
  cmp  rdi, rsi
  jg   .greater
  mov  rax, 0
  jmp .endif
.greater:
  mov rax, 1
.endif:
```

Conditionals

```
  mov  rdi, 3
  mov  rsi, 2
  cmp  rdi, rsi
  jg   .greater
  mov  rax, 0
  jmp .endif
.greater:
  mov rax, 1
.endif:
```

Conditionals

```
  mov  rdi, 2
  mov  rsi, 3
  cmp  rdi, rsi
  jg   .greater
  mov  rax, 0
  jmp .endif
.greater:
  mov rax, 1
.endif:
```

Conditionals

```
  mov  rdi, 2
  mov  rsi, 3
  cmp  rdi, rsi
  jg   .greater
  mov  rax, 0
  jmp .endif
.greater:
  mov rax, 1
.endif:
```

Conditionals

```
  mov  rdi, 2
  mov  rsi, 3
  cmp  rdi, rsi
  jg   .greater
  mov  rax, 0
  jmp .endif
.greater:
  mov rax, 1
.endif:
```

Conditionals

```
  mov  rdi, 2
  mov  rsi, 3
  cmp  rdi, rsi
  jg   .greater
  mov  rax, 0
  jmp .endif
.greater:
  mov rax, 1
.endif:
```

Conditionals

```
  mov  rdi, 2
  mov  rsi, 3
  cmp  rdi, rsi
  jg   .greater
  mov  rax, 0
  jmp .endif
.greater:
  mov rax, 1
.endif:
```

Conditionals

```
  mov  rdi, 2
  mov  rsi, 3
  cmp  rdi, rsi
  jg   .greater
  mov  rax, 0
  jmp .endif
.greater:
  mov rax, 1
.endif:
```

Conditionals

```
  mov  rdi, 2
  mov  rsi, 3
  cmp  rdi, rsi
  jg   .greater
  mov  rax, 0
  jmp .endif
.greater:
  mov rax, 1
.endif:
```

Jump around!

```
cmp op1 op2

je  label;  op1 == op2
jne label;  op1 != op2

jl  label;  op1 <  op2
jle label;  op1 <= op2
jg  label;  op1 >  op2
jge label;  op1 >= op2

jb  label;  op1 <  op2
jbe label;  op1 <= op2
ja  label;  op1 >  op2
jae label;  op1 >= op2
```

Loops

```
  mov rcx, 1
  mov rax, 0
.for
  cmp rcx, 3
  je  .endfor
  add rax, rcx
  inc rcx
  jmp .for
.endfor
```

Loops

```
  mov rcx, 1
  mov rax, 0
.for
  cmp rcx, 3
  je  .endfor
  add rax, rcx
  inc rcx
  jmp .for
.endfor
```

Loops

```
  mov rcx, 1
  mov rax, 0
.for
  cmp rcx, 3
  je  .endfor
  add rax, rcx
  inc rcx
  jmp .for
.endfor
```

Loops

```
  mov rcx, 1
  mov rax, 0
.for
  cmp rcx, 3
  je  .endfor
  add rax, rcx
  inc rcx
  jmp .for
.endfor
```

Loops

```
  mov rcx, 1
  mov rax, 0
.for
  cmp rcx, 3
  je  .endfor
  add rax, rcx
  inc rcx
  jmp .for
.endfor
```

Loops

```
  mov rcx, 1
  mov rax, 0
.for
  cmp rcx, 3
  je  .endfor
  add rax, rcx
  inc rcx
  jmp .for
.endfor
```

Loops

```
  mov rcx, 1
  mov rax, 0
.for
  cmp rcx, 3
  je  .endfor
  add rax, rcx
  inc rcx
  jmp .for
.endfor
```

Loops

```
  mov rcx, 1
  mov rax, 0
.for
  cmp rcx, 3
  je  .endfor
  add rax, rcx
  inc rcx
  jmp .for
.endfor
```

Loops

```
  mov rcx, 1
  mov rax, 0
.for
  cmp rcx, 3
  je  .endfor
  add rax, rcx
  inc rcx
  jmp .for
.endfor
```

Loops

```
  mov rcx, 1
  mov rax, 0
.for
  cmp rcx, 3
  je  .endfor
  add rax, rcx
  inc rcx
  jmp .for
.endfor
```

Loops

```
  mov rcx, 1
  mov rax, 0
.for
  cmp rcx, 3
  je  .endfor
  add rax, rcx
  inc rcx
  jmp .for
.endfor
```

Loops

```
  mov rcx, 1
  mov rax, 0
.for
  cmp rcx, 3
  je  .endfor
  add rax, rcx
  inc rcx
  jmp .for
.endfor
```

rax	0000000000000003
rcx	0000000000000003

Loops

```
  mov rcx, 1
  mov rax, 0
.for
  cmp rcx, 3
  je  .endfor
  add rax, rcx
  inc rcx
  jmp .for
.endfor
```

Loops

```
  mov rcx, 1
  mov rax, 0
.for
  cmp rcx, 3
  je  .endfor
  add rax, rcx
  inc rcx
  jmp .for
.endfor
```

Loops

```
  mov rcx, 1
  mov rax, 0
.for
  cmp rcx, 3
  je  .endfor
  add rax, rcx
  inc rcx
  jmp .for
.endfor
```

Loops

```
int rax = 0;
for (int rcx = 1; rcx != 3; ++rcx)
{
    rax += rcx;
}
```

```
  mov rcx, 1
  mov rax, 0
.for
  cmp rcx, 3
  je  .endfor
  add rax, rcx
  inc rcx
  jmp .for
.endfor
```

Real example

https://godbolt.org/z/18GzzqzYT

A complete function

```
push rbp
mov  rbp, rsp
mov  dword ptr [rbp-4], edi
mov  dword ptr [rbp-8], esi
mov  edx, dword ptr [rbp-4]
mov  eax, dword ptr [rbp-8]
add  eax, edx
pop  rbp
ret
```

rdi	0000000000000002
rsi	0000000000000003
rax	0000000000000000
rdx	0000000000000000
rsp	00000000000000b0
rbp	0000000000000000

	…00b3	00
	…00b2	00
	…00b1	00
rsp→	…00b0	00
	…00af	00
	…00ae	00
	…00ad	00
	…00ac	00
	…00ab	00
	…00aa	00
	…00a9	00
	…00a8	00
	…00a7	00
	…00a6	00
	…00a5	00
	…00a4	00
	…00a3	00
	…00a2	00
	…00a1	00
	…00a0	00

A complete function

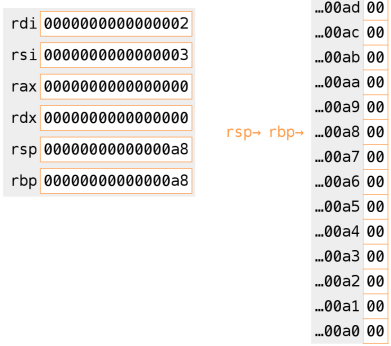
```
push rbp
mov  rbp, rsp
mov  dword ptr [rbp-4], edi
mov  dword ptr [rbp-8], esi
mov  edx, dword ptr [rbp-4]
mov  eax, dword ptr [rbp-8]
add  eax, edx
pop  rbp
ret
```

rdi	0000000000000002
rsi	0000000000000003
rax	0000000000000000
rdx	0000000000000000
rsp	00000000000000a8
rbp	0000000000000000

	…00b3	00
	…00b2	00
	…00b1	00
	…00b0	00
	…00af	00
	…00ae	00
	…00ad	00
	…00ac	00
	…00ab	00
	…00aa	00
	…00a9	00
rsp→	…00a8	00
	…00a7	00
	…00a6	00
	…00a5	00
	…00a4	00
	…00a3	00
	…00a2	00
	…00a1	00
	…00a0	00

A complete function

```
push rbp
mov  rbp, rsp
mov  dword ptr [rbp-4], edi
mov  dword ptr [rbp-8], esi
mov  edx, dword ptr [rbp-4]
mov  eax, dword ptr [rbp-8]
add  eax, edx
pop  rbp
ret
```



A complete function

```
push rbp
mov  rbp, rsp
mov  dword ptr [rbp-4], edi
mov  dword ptr [rbp-8], esi
mov  edx, dword ptr [rbp-4]
mov  eax, dword ptr [rbp-8]
add  eax, edx
pop  rbp
ret
```

rdi	0000000000000002
rsi	0000000000000003
rax	0000000000000000
rdx	0000000000000000
rsp	00000000000000a8
rbp	00000000000000a8

	…00b3	00
	…00b2	00
	…00b1	00
	…00b0	00
	…00af	00
	…00ae	00
	…00ad	00
	…00ac	00
	…00ab	00
	…00aa	00
	…00a9	00
rsp→ rbp→	…00a8	00
	…00a7	00
	…00a6	00
	…00a5	00
	…00a4	02
	…00a3	00
	…00a2	00
	…00a1	00
	…00a0	00

A complete function

```
push rbp
mov  rbp, rsp
mov  dword ptr [rbp-4], edi
mov  dword ptr [rbp-8], esi
mov  edx, dword ptr [rbp-4]
mov  eax, dword ptr [rbp-8]
add  eax, edx
pop  rbp
ret
```

rdi	0000000000000002
rsi	0000000000000003
rax	0000000000000000
rdx	0000000000000000
rsp	00000000000000a8
rbp	00000000000000a8

…00b3	00	
…00b2	00	
…00b1	00	
…00b0	00	
…00af	00	
…00ae	00	
…00ad	00	
…00ac	00	
…00ab	00	
…00aa	00	
…00a9	00	
…00a8	00	rsp→ rbp→
…00a7	00	
…00a6	00	
…00a5	00	
…00a4	02	
…00a3	00	
…00a2	00	
…00a1	00	
…00a0	03	

A complete function

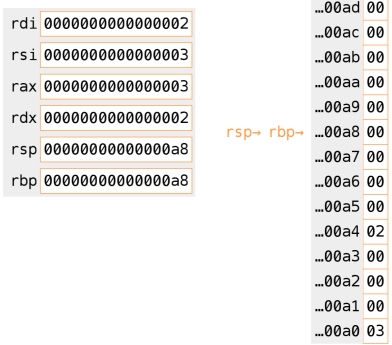
```
push rbp
mov  rbp, rsp
mov  dword ptr [rbp-4], edi
mov  dword ptr [rbp-8], esi
mov  edx, dword ptr [rbp-4]
mov  eax, dword ptr [rbp-8]
add  eax, edx
pop  rbp
ret
```

rdi	0000000000000002
rsi	0000000000000003
rax	0000000000000000
rdx	0000000000000002
rsp	00000000000000a8
rbp	00000000000000a8

	…00b3	00
	…00b2	00
	…00b1	00
	…00b0	00
	…00af	00
	…00ae	00
	…00ad	00
	…00ac	00
	…00ab	00
	…00aa	00
	…00a9	00
rsp→ rbp→	…00a8	00
	…00a7	00
	…00a6	00
	…00a5	00
	…00a4	02
	…00a3	00
	…00a2	00
	…00a1	00
	…00a0	03

A complete function

```
push rbp
mov  rbp, rsp
mov  dword ptr [rbp-4], edi
mov  dword ptr [rbp-8], esi
mov  edx, dword ptr [rbp-4]
mov  eax, dword ptr [rbp-8]
add  eax, edx
pop  rbp
ret
```



A complete function

```
push rbp
mov  rbp, rsp
mov  dword ptr [rbp-4], edi
mov  dword ptr [rbp-8], esi
mov  edx, dword ptr [rbp-4]
mov  eax, dword ptr [rbp-8]
add  eax, edx
pop  rbp
ret
```

rdi	0000000000000002
rsi	0000000000000003
rax	0000000000000005
rdx	0000000000000002
rsp	00000000000000a8
rbp	00000000000000a8

	…00b3	00
	…00b2	00
	…00b1	00
	…00b0	00
	…00af	00
	…00ae	00
	…00ad	00
	…00ac	00
	…00ab	00
	…00aa	00
	…00a9	00
rsp→ rbp→	…00a8	00
	…00a7	00
	…00a6	00
	…00a5	00
	…00a4	02
	…00a3	00
	…00a2	00
	…00a1	00
	…00a0	03

A complete function

```
push rbp
mov  rbp, rsp
mov  dword ptr [rbp-4], edi
mov  dword ptr [rbp-8], esi
mov  edx, dword ptr [rbp-4]
mov  eax, dword ptr [rbp-8]
add  eax, edx
pop  rbp
ret
```

rdi	0000000000000002
rsi	0000000000000003
rax	0000000000000005
rdx	0000000000000002
rsp	00000000000000b0
rbp	0000000000000000

	…00b3	00
	…00b2	00
	…00b1	00
rsp→	…00b0	00
	…00af	00
	…00ae	00
	…00ad	00
	…00ac	00
	…00ab	00
	…00aa	00
	…00a9	00
	…00a8	00
	…00a7	00
	…00a6	00
	…00a5	00
	…00a4	02
	…00a3	00
	…00a2	00
	…00a1	00
	…00a0	03

A complete function

```
push rbp
mov  rbp, rsp
mov  dword ptr [rbp-4], edi
mov  dword ptr [rbp-8], esi
mov  edx, dword ptr [rbp-4]
mov  eax, dword ptr [rbp-8]
add  eax, edx
pop  rbp
ret
```

rdi	0000000000000002
rsi	0000000000000003
rax	0000000000000005
rdx	0000000000000002
rsp	00000000000000b0
rbp	0000000000000000

	…00b3	00
	…00b2	00
	…00b1	00
rsp→	…00b0	00
	…00af	00
	…00ae	00
	…00ad	00
	…00ac	00
	…00ab	00
	…00aa	00
	…00a9	00
	…00a8	00
	…00a7	00
	…00a6	00
	…00a5	00
	…00a4	02
	…00a3	00
	…00a2	00
	…00a1	00
	…00a0	03

Optimization

https://godbolt.org/z/18GzzqzYT

Function call

https://godbolt.org/z/8zaM3Eeb7

Function call

```
sum_times_two_int_int_:
  push rbp
  mov  rbp, rsp
  sub  rsp, 16
  mov  dword ptr [rbp-4], edi
  mov  dword ptr [rbp-8], esi
  mov  edx, dword ptr [rbp-4]
  mov  eax, dword ptr [rbp-8]
  add  eax, edx
  mov  edi, eax
  call times_two_int_
  leave
  ret
times_two_int_:
  push rbp
  mov  rbp, rsp
  mov  dword ptr [rbp-4], edi
  mov  eax, dword ptr [rbp-4]
  add  eax, eax
  pop  rbp
  ret
```

rdi	0000000000000002
rsi	0000000000000003
rax	0000000000000000
rdx	0000000000000000
rbp	0000000000000000
rsp	00000000000000b0

	…00b3	00		…009f	00
	…00b2	00		…009e	00
	…00b1	00		…009d	00
rsp→	…00b0	00		…009c	00
	…00af	00		…009b	00
	…00ae	00		…009a	00
	…00ad	00		…0099	00
	…00ac	00		…0098	00
	…00ab	00		…0097	00
	…00aa	00		…0096	00
	…00a9	00		…0095	00
	…00a8	00		…0094	00
	…00a7	00		…0093	00
	…00a6	00		…0092	00
	…00a5	00		…0091	00
	…00a4	00		…0090	00
	…00a3	00		…008f	00
	…00a2	00		…008e	00
	…00a1	00		…008d	00
	…00a0	00		…008c	00

Function call

```
sum_times_two_int_int_:
  push rbp
  mov  rbp, rsp
  sub  rsp, 16
  mov  dword ptr [rbp-4], edi
  mov  dword ptr [rbp-8], esi
  mov  edx, dword ptr [rbp-4]
  mov  eax, dword ptr [rbp-8]
  add  eax, edx
  mov  edi, eax
  call times_two_int_
  leave
  ret
times_two_int_:
  push rbp
  mov  rbp, rsp
  mov  dword ptr [rbp-4], edi
  mov  eax, dword ptr [rbp-4]
  add  eax, eax
  pop  rbp
  ret
```

rdi	0000000000000002
rsi	0000000000000003
rax	0000000000000000
rdx	0000000000000000
rbp	0000000000000000
rsp	00000000000000a8

	…00b3	00	…009f 00
	…00b2	00	…009e 00
	…00b1	00	…009d 00
	…00b0	00	…009c 00
	…00af	00	…009b 00
	…00ae	00	…009a 00
	…00ad	00	…0099 00
	…00ac	00	…0098 00
	…00ab	00	…0097 00
	…00aa	00	…0096 00
	…00a9	00	…0095 00
rsp→	…00a8	00	…0094 00
	…00a7	00	…0093 00
	…00a6	00	…0092 00
	…00a5	00	…0091 00
	…00a4	00	…0090 00
	…00a3	00	…008f 00
	…00a2	00	…008e 00
	…00a1	00	…008d 00
	…00a0	00	…008c 00

Function call

```
sum_times_two_int_int_:
  push rbp
  mov  rbp, rsp
  sub  rsp, 16
  mov  dword ptr [rbp-4], edi
  mov  dword ptr [rbp-8], esi
  mov  edx, dword ptr [rbp-4]
  mov  eax, dword ptr [rbp-8]
  add  eax, edx
  mov  edi, eax
  call times_two_int_
  leave
  ret
times_two_int_:
  push rbp
  mov  rbp, rsp
  mov  dword ptr [rbp-4], edi
  mov  eax, dword ptr [rbp-4]
  add  eax, eax
  pop  rbp
  ret
```

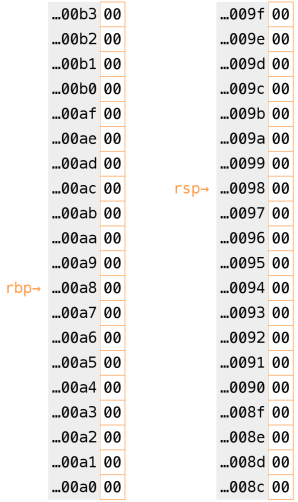
rdi	0000000000000002
rsi	0000000000000003
rax	0000000000000000
rdx	0000000000000000
rbp	00000000000000a8
rsp	00000000000000a8

	…00b3	00	…009f 00
	…00b2	00	…009e 00
	…00b1	00	…009d 00
	…00b0	00	…009c 00
	…00af	00	…009b 00
	…00ae	00	…009a 00
	…00ad	00	…0099 00
	…00ac	00	…0098 00
	…00ab	00	…0097 00
	…00aa	00	…0096 00
	…00a9	00	…0095 00
rsp→ rbp→	…00a8	00	…0094 00
	…00a7	00	…0093 00
	…00a6	00	…0092 00
	…00a5	00	…0091 00
	…00a4	00	…0090 00
	…00a3	00	…008f 00
	…00a2	00	…008e 00
	…00a1	00	…008d 00
	…00a0	00	…008c 00

Function call

```
sum_times_two_int_int_:
  push rbp
  mov  rbp, rsp
  sub  rsp, 16
  mov  dword ptr [rbp-4], edi
  mov  dword ptr [rbp-8], esi
  mov  edx, dword ptr [rbp-4]
  mov  eax, dword ptr [rbp-8]
  add  eax, edx
  mov  edi, eax
  call times_two_int_
  leave
  ret
times_two_int_:
  push rbp
  mov  rbp, rsp
  mov  dword ptr [rbp-4], edi
  mov  eax, dword ptr [rbp-4]
  add  eax, eax
  pop  rbp
  ret
```

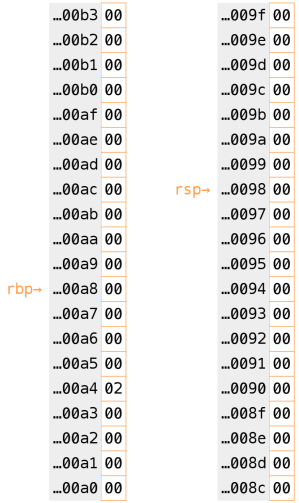
rdi	0000000000000002
rsi	0000000000000003
rax	0000000000000000
rdx	0000000000000000
rbp	00000000000000a8
rsp	0000000000000098



Function call

```
sum_times_two_int_int_:
  push rbp
  mov  rbp, rsp
  sub  rsp, 16
  mov  dword ptr [rbp-4], edi
  mov  dword ptr [rbp-8], esi
  mov  edx, dword ptr [rbp-4]
  mov  eax, dword ptr [rbp-8]
  add  eax, edx
  mov  edi, eax
  call times_two_int_
  leave
  ret
times_two_int_:
  push rbp
  mov  rbp, rsp
  mov  dword ptr [rbp-4], edi
  mov  eax, dword ptr [rbp-4]
  add  eax, eax
  pop  rbp
  ret
```

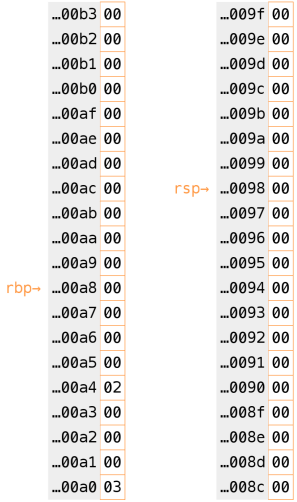
rdi	0000000000000002
rsi	0000000000000003
rax	0000000000000000
rdx	0000000000000000
rbp	00000000000000a8
rsp	0000000000000098



Function call

```
sum_times_two_int_int_:
  push rbp
  mov  rbp, rsp
  sub  rsp, 16
  mov  dword ptr [rbp-4], edi
  mov  dword ptr [rbp-8], esi
  mov  edx, dword ptr [rbp-4]
  mov  eax, dword ptr [rbp-8]
  add  eax, edx
  mov  edi, eax
  call times_two_int_
  leave
  ret
times_two_int_:
  push rbp
  mov  rbp, rsp
  mov  dword ptr [rbp-4], edi
  mov  eax, dword ptr [rbp-4]
  add  eax, eax
  pop  rbp
  ret
```

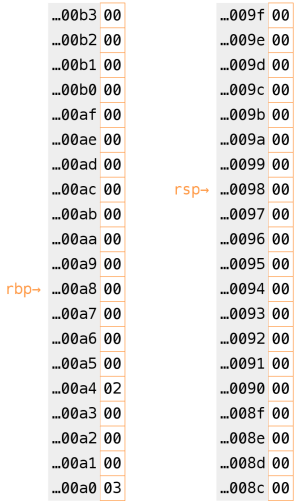
rdi	0000000000000002
rsi	0000000000000003
rax	0000000000000000
rdx	0000000000000000
rbp	00000000000000a8
rsp	0000000000000098



Function call

```
sum_times_two_int_int_:
  push rbp
  mov  rbp, rsp
  sub  rsp, 16
  mov  dword ptr [rbp-4], edi
  mov  dword ptr [rbp-8], esi
  mov  edx, dword ptr [rbp-4]
  mov  eax, dword ptr [rbp-8]
  add  eax, edx
  mov  edi, eax
  call times_two_int_
  leave
  ret
times_two_int_:
  push rbp
  mov  rbp, rsp
  mov  dword ptr [rbp-4], edi
  mov  eax, dword ptr [rbp-4]
  add  eax, eax
  pop  rbp
  ret
```

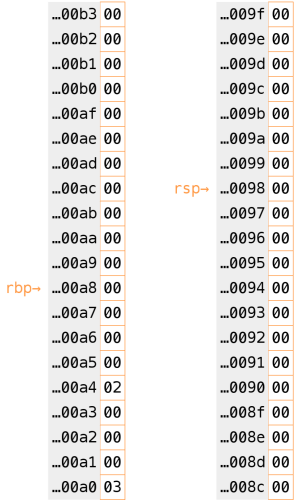
rdi	0000000000000002
rsi	0000000000000003
rax	0000000000000000
rdx	0000000000000002
rbp	00000000000000a8
rsp	0000000000000098



Function call

```
sum_times_two_int_int_:
  push rbp
  mov  rbp, rsp
  sub  rsp, 16
  mov  dword ptr [rbp-4], edi
  mov  dword ptr [rbp-8], esi
  mov  edx, dword ptr [rbp-4]
  mov  eax, dword ptr [rbp-8]
  add  eax, edx
  mov  edi, eax
  call times_two_int_
  leave
  ret
times_two_int_:
  push rbp
  mov  rbp, rsp
  mov  dword ptr [rbp-4], edi
  mov  eax, dword ptr [rbp-4]
  add  eax, eax
  pop  rbp
  ret
```

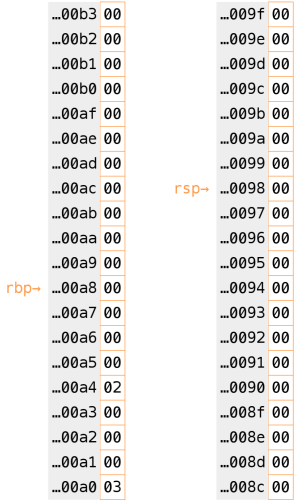
rdi	0000000000000002
rsi	0000000000000003
rax	0000000000000003
rdx	0000000000000002
rbp	00000000000000a8
rsp	0000000000000098



Function call

```
sum_times_two_int_int_:
  push rbp
  mov  rbp, rsp
  sub  rsp, 16
  mov  dword ptr [rbp-4], edi
  mov  dword ptr [rbp-8], esi
  mov  edx, dword ptr [rbp-4]
  mov  eax, dword ptr [rbp-8]
  add  eax, edx
  mov  edi, eax
  call times_two_int_
  leave
  ret
times_two_int_:
  push rbp
  mov  rbp, rsp
  mov  dword ptr [rbp-4], edi
  mov  eax, dword ptr [rbp-4]
  add  eax, eax
  pop  rbp
  ret
```

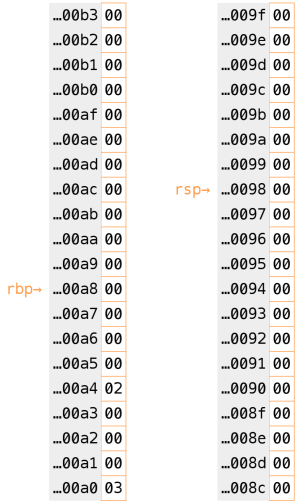
rdi	0000000000000002
rsi	0000000000000003
rax	0000000000000005
rdx	0000000000000002
rbp	00000000000000a8
rsp	0000000000000098



Function call

```
sum_times_two_int_int_:
  push rbp
  mov  rbp, rsp
  sub  rsp, 16
  mov  dword ptr [rbp-4], edi
  mov  dword ptr [rbp-8], esi
  mov  edx, dword ptr [rbp-4]
  mov  eax, dword ptr [rbp-8]
  add  eax, edx
  mov  edi, eax
  call times_two_int_
  leave
  ret
times_two_int_:
  push rbp
  mov  rbp, rsp
  mov  dword ptr [rbp-4], edi
  mov  eax, dword ptr [rbp-4]
  add  eax, eax
  pop  rbp
  ret
```

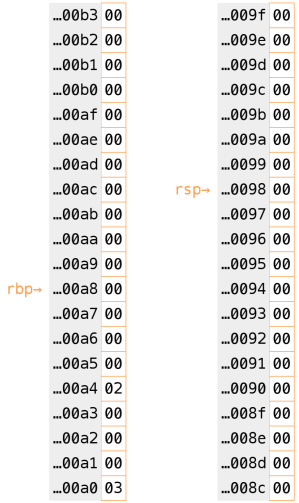
rdi	0000000000000005
rsi	0000000000000003
rax	0000000000000005
rdx	0000000000000002
rbp	00000000000000a8
rsp	0000000000000098



Function call

```
sum_times_two_int_int_:
  push rbp
  mov  rbp, rsp
  sub  rsp, 16
  mov  dword ptr [rbp-4], edi
  mov  dword ptr [rbp-8], esi
  mov  edx, dword ptr [rbp-4]
  mov  eax, dword ptr [rbp-8]
  add  eax, edx
  mov  edi, eax
  call times_two_int_
  leave
  ret
times_two_int_:
  push rbp
  mov  rbp, rsp
  mov  dword ptr [rbp-4], edi
  mov  eax, dword ptr [rbp-4]
  add  eax, eax
  pop  rbp
  ret
```

rdi	0000000000000005
rsi	0000000000000003
rax	0000000000000005
rdx	0000000000000002
rbp	00000000000000a8
rsp	0000000000000098



Function call

```
sum_times_two_int_int_:
  push rbp
  mov  rbp, rsp
  sub  rsp, 16
  mov  dword ptr [rbp-4], edi
  mov  dword ptr [rbp-8], esi
  mov  edx, dword ptr [rbp-4]
  mov  eax, dword ptr [rbp-8]
  add  eax, edx
  mov  edi, eax
  call times_two_int_
  leave
  ret
times_two_int_:
  push rbp
  mov  rbp, rsp
  mov  dword ptr [rbp-4], edi
  mov  eax, dword ptr [rbp-4]
  add  eax, eax
  pop  rbp
  ret
```

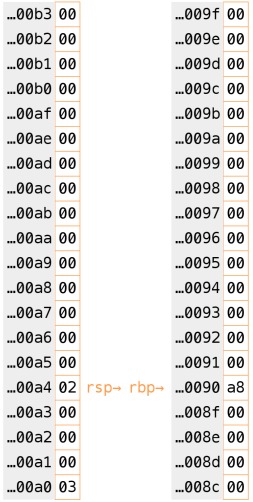
rdi	0000000000000005
rsi	0000000000000003
rax	0000000000000005
rdx	0000000000000002
rbp	00000000000000a8
rsp	0000000000000090

	…00b3	00		…009f 00
	…00b2	00		…009e 00
	…00b1	00		…009d 00
	…00b0	00		…009c 00
	…00af	00		…009b 00
	…00ae	00		…009a 00
	…00ad	00		…0099 00
	…00ac	00		…0098 00
	…00ab	00		…0097 00
	…00aa	00		…0096 00
	…00a9	00		…0095 00
rbp→	…00a8	00		…0094 00
	…00a7	00		…0093 00
	…00a6	00		…0092 00
	…00a5	00		…0091 00
	…00a4	02	rsp→	…0090 a8
	…00a3	00		…008f 00
	…00a2	00		…008e 00
	…00a1	00		…008d 00
	…00a0	03		…008c 00

Function call

```
sum_times_two_int_int_:
  push rbp
  mov  rbp, rsp
  sub  rsp, 16
  mov  dword ptr [rbp-4], edi
  mov  dword ptr [rbp-8], esi
  mov  edx, dword ptr [rbp-4]
  mov  eax, dword ptr [rbp-8]
  add  eax, edx
  mov  edi, eax
  call times_two_int_
  leave
  ret
times_two_int_:
  push rbp
  mov  rbp, rsp
  mov  dword ptr [rbp-4], edi
  mov  eax, dword ptr [rbp-4]
  add  eax, eax
  pop  rbp
  ret
```

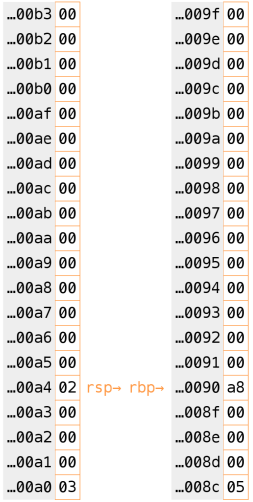
rdi	0000000000000005
rsi	0000000000000003
rax	0000000000000005
rdx	0000000000000002
rbp	0000000000000090
rsp	0000000000000090



Function call

```
sum_times_two_int_int_:
  push rbp
  mov  rbp, rsp
  sub  rsp, 16
  mov  dword ptr [rbp-4], edi
  mov  dword ptr [rbp-8], esi
  mov  edx, dword ptr [rbp-4]
  mov  eax, dword ptr [rbp-8]
  add  eax, edx
  mov  edi, eax
  call times_two_int_
  leave
  ret
times_two_int_:
  push rbp
  mov  rbp, rsp
  mov  dword ptr [rbp-4], edi
  mov  eax, dword ptr [rbp-4]
  add  eax, eax
  pop  rbp
  ret
```

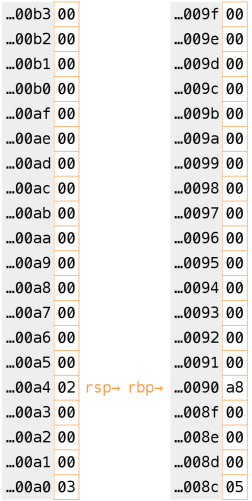
rdi	0000000000000005
rsi	0000000000000003
rax	0000000000000005
rdx	0000000000000002
rbp	0000000000000090
rsp	0000000000000090



Function call

```
sum_times_two_int_int_:
  push rbp
  mov  rbp, rsp
  sub  rsp, 16
  mov  dword ptr [rbp-4], edi
  mov  dword ptr [rbp-8], esi
  mov  edx, dword ptr [rbp-4]
  mov  eax, dword ptr [rbp-8]
  add  eax, edx
  mov  edi, eax
  call times_two_int_
  leave
  ret
times_two_int_:
  push rbp
  mov  rbp, rsp
  mov  dword ptr [rbp-4], edi
  mov  eax, dword ptr [rbp-4]
  add  eax, eax
  pop  rbp
  ret
```

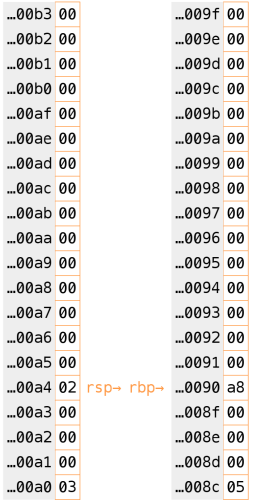
rdi	0000000000000005
rsi	0000000000000003
rax	0000000000000005
rdx	0000000000000002
rbp	0000000000000090
rsp	0000000000000090



Function call

```
sum_times_two_int_int_:
  push rbp
  mov  rbp, rsp
  sub  rsp, 16
  mov  dword ptr [rbp-4], edi
  mov  dword ptr [rbp-8], esi
  mov  edx, dword ptr [rbp-4]
  mov  eax, dword ptr [rbp-8]
  add  eax, edx
  mov  edi, eax
  call times_two_int_
  leave
  ret
times_two_int_:
  push rbp
  mov  rbp, rsp
  mov  dword ptr [rbp-4], edi
  mov  eax, dword ptr [rbp-4]
  add  eax, eax
  pop  rbp
  ret
```

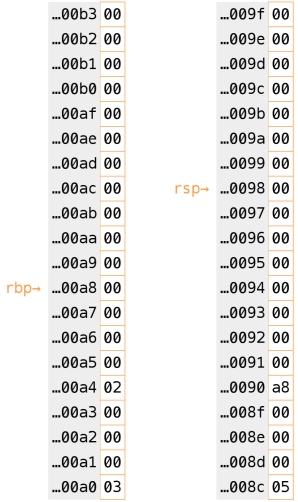
rdi	0000000000000005
rsi	0000000000000003
rax	000000000000000a
rdx	0000000000000002
rbp	0000000000000090
rsp	0000000000000090



Function call

```
sum_times_two_int_int_:
  push rbp
  mov  rbp, rsp
  sub  rsp, 16
  mov  dword ptr [rbp-4], edi
  mov  dword ptr [rbp-8], esi
  mov  edx, dword ptr [rbp-4]
  mov  eax, dword ptr [rbp-8]
  add  eax, edx
  mov  edi, eax
  call times_two_int_
  leave
  ret
times_two_int_:
  push rbp
  mov  rbp, rsp
  mov  dword ptr [rbp-4], edi
  mov  eax, dword ptr [rbp-4]
  add  eax, eax
  pop  rbp
  ret
```

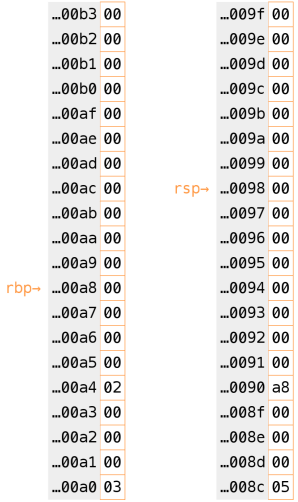
rdi	0000000000000005
rsi	0000000000000003
rax	000000000000000a
rdx	0000000000000002
rbp	00000000000000a8
rsp	0000000000000098



Function call

```
sum_times_two_int_int_:
  push rbp
  mov  rbp, rsp
  sub  rsp, 16
  mov  dword ptr [rbp-4], edi
  mov  dword ptr [rbp-8], esi
  mov  edx, dword ptr [rbp-4]
  mov  eax, dword ptr [rbp-8]
  add  eax, edx
  mov  edi, eax
  call times_two_int_
  leave
  ret
times_two_int_:
  push rbp
  mov  rbp, rsp
  mov  dword ptr [rbp-4], edi
  mov  eax, dword ptr [rbp-4]
  add  eax, eax
  pop  rbp
  ret
```

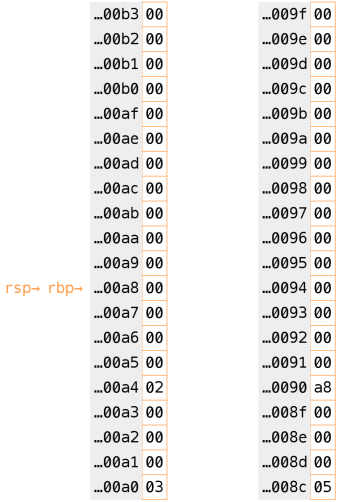
rdi	0000000000000005
rsi	0000000000000003
rax	000000000000000a
rdx	0000000000000002
rbp	00000000000000a8
rsp	0000000000000098



Function call

```
sum_times_two_int_int_:
  push rbp
  mov  rbp, rsp
  sub  rsp, 16
  mov  dword ptr [rbp-4], edi
  mov  dword ptr [rbp-8], esi
  mov  edx, dword ptr [rbp-4]
  mov  eax, dword ptr [rbp-8]
  add  eax, edx
  mov  edi, eax
  call times_two_int_
  leave
  ret
times_two_int_:
  push rbp
  mov  rbp, rsp
  mov  dword ptr [rbp-4], edi
  mov  eax, dword ptr [rbp-4]
  add  eax, eax
  pop  rbp
  ret
```

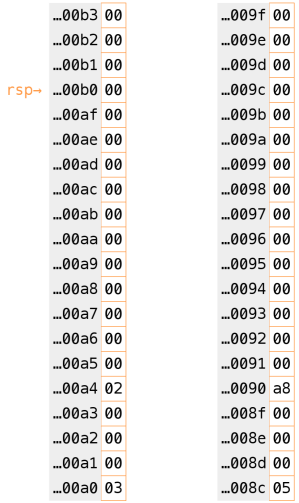
rdi	0000000000000005
rsi	0000000000000003
rax	000000000000000a
rdx	0000000000000002
rbp	00000000000000a8
rsp	00000000000000a8



Function call

```
sum_times_two_int_int_:
  push rbp
  mov  rbp, rsp
  sub  rsp, 16
  mov  dword ptr [rbp-4], edi
  mov  dword ptr [rbp-8], esi
  mov  edx, dword ptr [rbp-4]
  mov  eax, dword ptr [rbp-8]
  add  eax, edx
  mov  edi, eax
  call times_two_int_
  leave
  ret
times_two_int_:
  push rbp
  mov  rbp, rsp
  mov  dword ptr [rbp-4], edi
  mov  eax, dword ptr [rbp-4]
  add  eax, eax
  pop  rbp
  ret
```

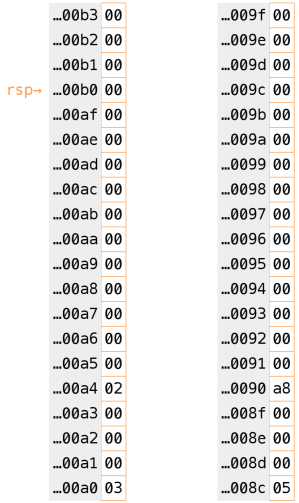
rdi	0000000000000005
rsi	0000000000000003
rax	000000000000000a
rdx	0000000000000002
rbp	0000000000000000
rsp	00000000000000b0



Function call

```
sum_times_two_int_int_:
  push rbp
  mov  rbp, rsp
  sub  rsp, 16
  mov  dword ptr [rbp-4], edi
  mov  dword ptr [rbp-8], esi
  mov  edx, dword ptr [rbp-4]
  mov  eax, dword ptr [rbp-8]
  add  eax, edx
  mov  edi, eax
  call times_two_int_
  leave
  ret
times_two_int_:
  push rbp
  mov  rbp, rsp
  mov  dword ptr [rbp-4], edi
  mov  eax, dword ptr [rbp-4]
  add  eax, eax
  pop  rbp
  ret
```

rdi	0000000000000005
rsi	0000000000000003
rax	000000000000000a
rdx	0000000000000002
rbp	0000000000000000
rsp	00000000000000b0



Function call

https://godbolt.org/z/8zaM3Eeb7

Compiler Explorer

https://godbolt.org/z/qTeo75jrr

Tips

- Hover instruction, then google

Tips

- Hover instruction, then google
- Make local standalone reproduction

Tips

- Hover instruction, then google
- Make local standalone reproduction
- Compiler Explorer has libraries!

Tips

- Hover instruction, then google
- Make local standalone reproduction
- Compiler Explorer has libraries!
- Use ints instead of complicated types

Tips

- Hover instruction, then google
- Make local standalone reproduction
- Compiler Explorer has libraries!
- Use ints instead of complicated types
- Don't try it on your full project

Just Enough Assembly for Compiler Explorer

Anders Schau Knatten (@knatten)

Zivid / CppQuiz

2021-12-01