От Ассемблера к C++

11 февраля 2026 г.

myexit.s

```
.global  _start
.section  .text
_start:
        movq  $60, %rax
        movq  $3, %rdi
        syscall
```

Построение программы из исходного кода

```
as myexit.s -o myexit.o
ld myexit.o -o myexit
```

Арифметические инструкции

- addq
- subq
- incq
- decq
- mulq
- divq

arithmetic.s

```
.global  _start
         .section  .text
         _start:
                  # Perform various arithmetic
                  # functions
                  movq $3, %rdi
                  movq %rdi, %rax
                  addq %rdi, %rax
                  mulq %rdi
                  movq %rax, %rdi
                  # Set the exit system
                  # call numberf
                  movq $60, %rax
                  # Perform the system call
                  movq $3, %rdi
                  syscall
```

Регистры

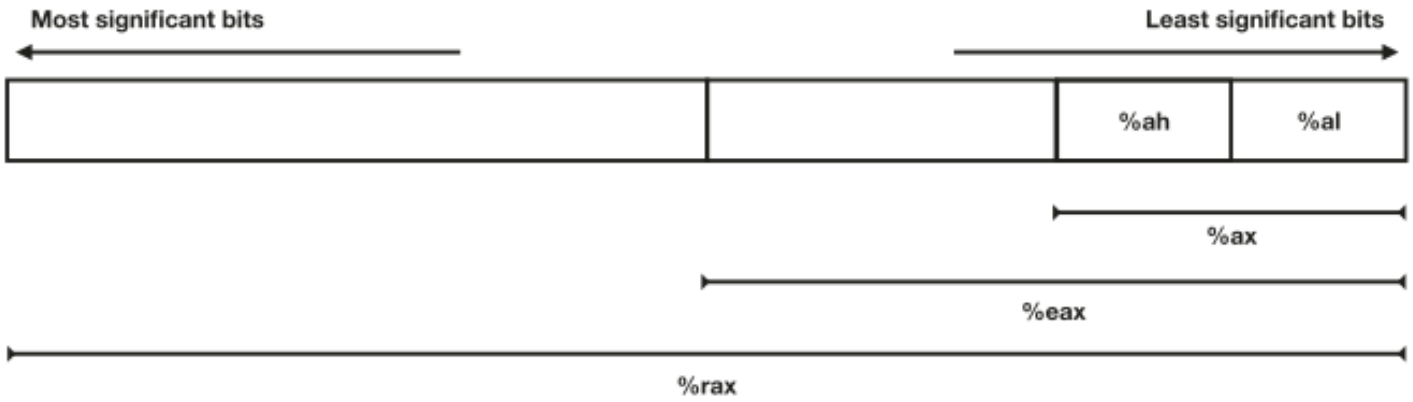


Рис.: Части регистра rax

Регистры общего назначения

16 регистров общего назначения.

- %rax: Аккумулятор.
- %rbx: регистр базы. Обчтно используется для индексиорованной адресации.
- %rcx: регистр счета. Используется для счета когда выполняется цикл.
- %rdx: регист данных.

Регистры общего назначения

- %rsi: Указатель источника. Часто для работы с большими участаками памяти.
- %rdi: Указатель назначения. Часто используется в сочетании с %rsi.
- %rbp: Указатель базы.
- %rsp: Указатель стэка.

Регистры общего назначения

8 регистров x86-64 ISA. %r8 - %r15.

jmp интсрукция

```
.global _start
.section .text
_start:
        movq $7, %rdi
        jmp nextplace

        # these two instructions are skipped
        movq $8, %rbx
        addq %rbx, %rdi

nextplace:
        movq $60, %rax
        syscall
```

Условные переходы и %eflags регистр

- zf: zero flag 1 если результат последней арефметической операции 0, 0 в противном случае.
- cf: carry flag 1 если результат последней арефметической операции больше чем должен быть в регистре назначения.

Условные переходы и %eflags регистр

- jz: Jump if Zero.
- jnz: Jump if Not Zero.
- jc: Jump if Carry.
- jnc: Jump if No Carry.

exponent.s

```
.global _start

# this will calculate 2^3.
# You can modify %rbx and %rcx to calculate
# another exponential.

.section .text
_start:
        # %rbx will hold the base
        movq $2, %rbx

        # %rcx will hold the current expoent count
        movq $3, %rcx

        # Store the accumluated value in %rax
        movq $1, %rax
```

exponent.s

```
mainloop:
        # Adding zero will allow us to use the flags t
        # determine if %rcx haz zero to begin with
        addq $0, %rcx

        # If the exponent is zero, we are done
        jz complete

        # Otherwise, mulitply the accumulated value by
        mulq %rbx

        # Decrease the counter
        decq %rcx

        # Go back to the begining of the loop and try
        jmp mainloop
```

exponent.s

```
complete:
        # Move the accumulated value to %rdi so we can
        movq %rax, %rdi
        # call the exit system call
        movq $60, %rax
        syscall
```

Сравнение

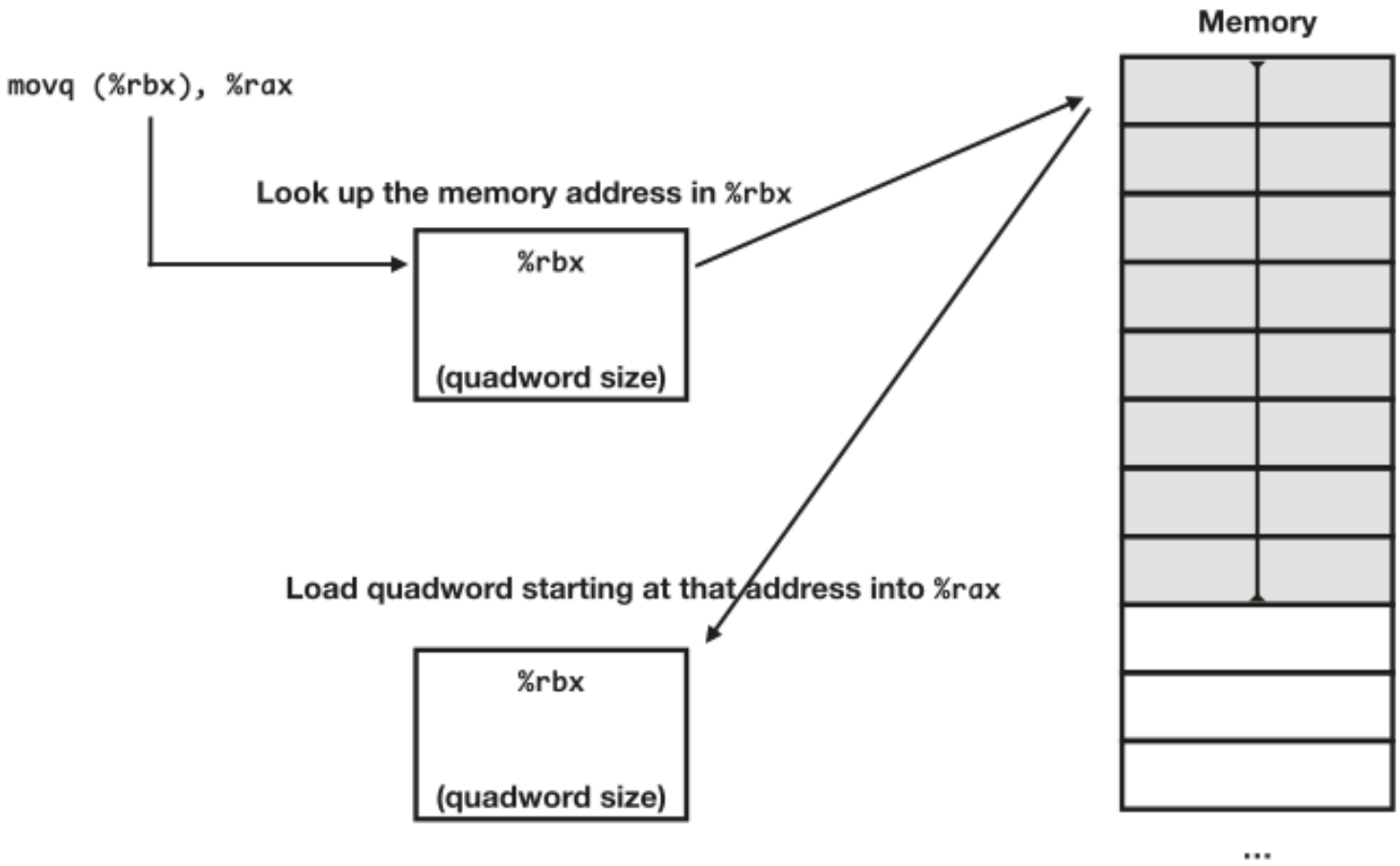
cmpq(cmpb, cmpw, cmpl) иснтрукция.

- je перейти, если arg2 равен arg1.
- jne.
- ja перейти, если arg2 (больше) above arg1.
- jae.
- jb перейти, если arg2 меньше(below) arg1.

Моды аресации памяти

- Immediate mode: movq $5, %rax. $ указывает на immediate mode.
- Register mode: movq $5, %rax. %rax - регистр.
- Direct memory mode: movq firstValue, %rbx. firstValue - это прямой адрес памяти.

Register Indirect addressing mode



Общая мода аресации

- value(basereg, idxreg, multipier)
- address = value + basereg + idxreg * multiplier

Общая мода аресации