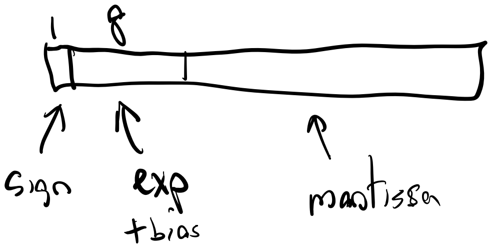
IEEE FP



If the number is $-1.\underline{011} \times 2^{15}$

Sign bit: 1

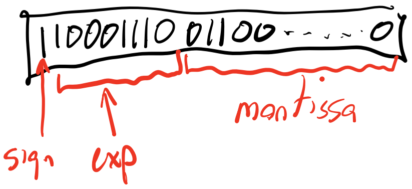
Exponent bits: $15 + 127 = 142$ decimal

$= 128 + 8 + 4 + 2$.

$= 10001110$ binary

Mantissa bits $= \underbrace{\underline{011}\,\underbrace{000\cdots0}_{\text{20 zero's}}}_{\text{23 bits}}$

(dropped 1 before the point)



What does 110.0101 mean?

4s 2s 1s $2^{-1} = 1/2$ $2^{-2} = 1/4$ 1/8 1/16

$= 4 + 2 + 0.25 + .0625 = 6.3125$

The computer's "Architecture"

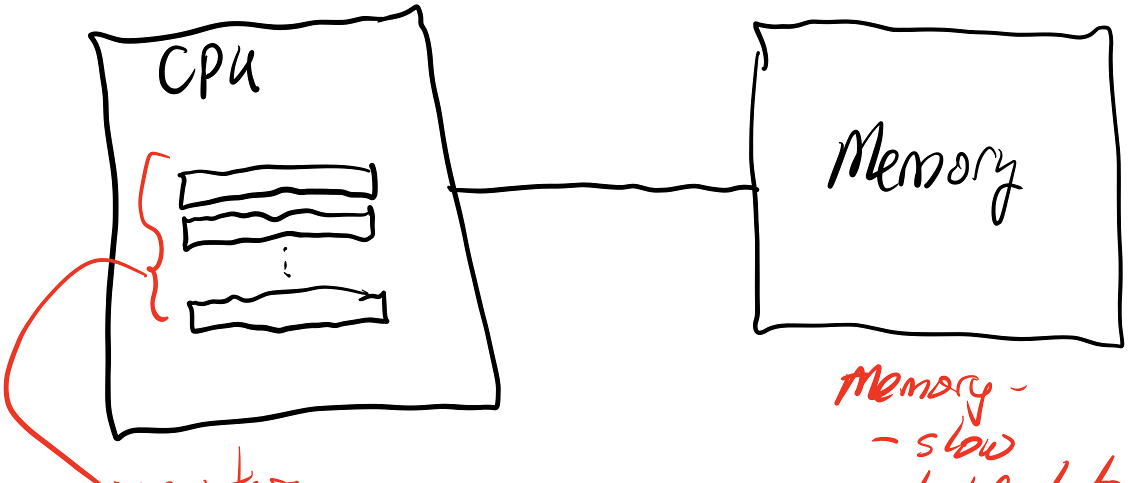
- how the ardware looks to an assembly language programmer.
  - assembly language is the human readable version of machine code.
- exposes the machine instructions and the memory to the programmer.

aka "Instruction Set Architecture" (ISA)

"Microarchitecture" - the design of the circuits that make up the computer.

Assembly Language

- consists of simple operations (instructions)
  - add, subtract, multiply, compare, jump, move data, AND, OR, XOR, etc.
- data operated on is in memory or in registers.



registers
- very fast
- small number of registers
- accessed by name.

Memory -
- slow
- holds lots of data
- accessed via addresses.

Intel x86-64 Architecture

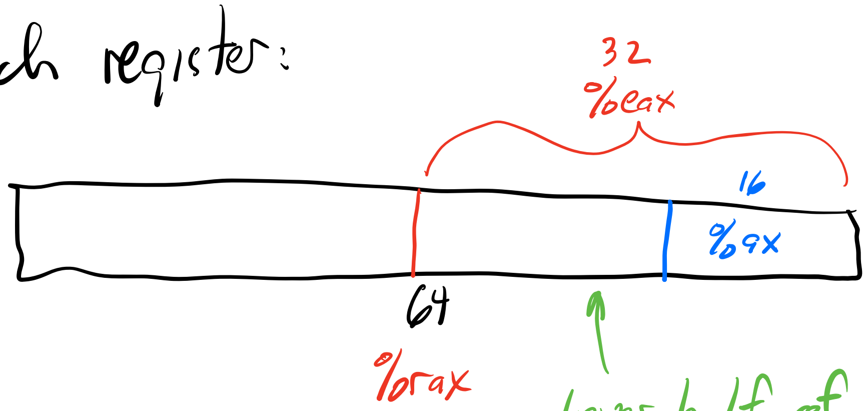
- 64-bit architecture
- aka "x64"
- we'll be programming in x64 assembly.

The x64 registers

- The general-purpose registers
    - each 64-bits
        - can one 64-bit pointer
        - or one 64-bit integer
            - can also hold smaller integers (32, 16, 8 bit)
- There are 16 of these registers:
  %rax, %rbx, %rcx, %rdx, %rsi,
  %rdi, %r8, %r9, ..., %r15
    - %rsp, %rbp } special purpose, leave alone.

    - see "cheat sheet" on Brightspace

For each register:



lower half of %r8..%r15 are referred to as %r8d...%r15d

Instructions

Move instruction

```
mov   source, destination      # copies from
                               # source to destination

mov %rcx, %rsi                 # copies %rcx into
                               # %rsi
```

The source can be a constant

mov $\underbrace{\$23}_{\text{Constant 23}}$, %rdx

Either the source or destination can be a memory address.

– but not both

Arithmetic instructions

only one of these can be a memory address.

```
add  source, destination
                #dest += source

sub   s,d    # d = d - s
imul  s,d    # d = d*s   (integer mult.)
inc   d      # d++
dec   d      # d--
and   s,d    # d = d & s
or    s,d    # d = d | s
...
```

Examples

```
add $3, %rax        #%rax += 3
sub %rcx, %rdx      #%rdx -= %rcx
```

Comparison operation

```
cmp  op2,op1   # compares op1 to
               # op2
               # (reversed)
```

-hardware remembers the result of the comparison.

jump operations

```
jmp label   # always jump to the label
```

Conditional jumps - come after cmp

```
jg label   # jump if the result of the
             comparison was "greater"
           # don't jump other wise
je    # jump on equal
jge   # jump on >=
jl    # jump on <
jle   # jump on <=
```

- often append "q" at the end of a 64-bit instruction
- "l" at the end of a 32-bit instruction
- usually optional
  - only required if assembler can't tell whether a 64-bit or 32-bit operation is intended.

Examples

```
movq %rax, %rcx
addl $52, %r8d
```