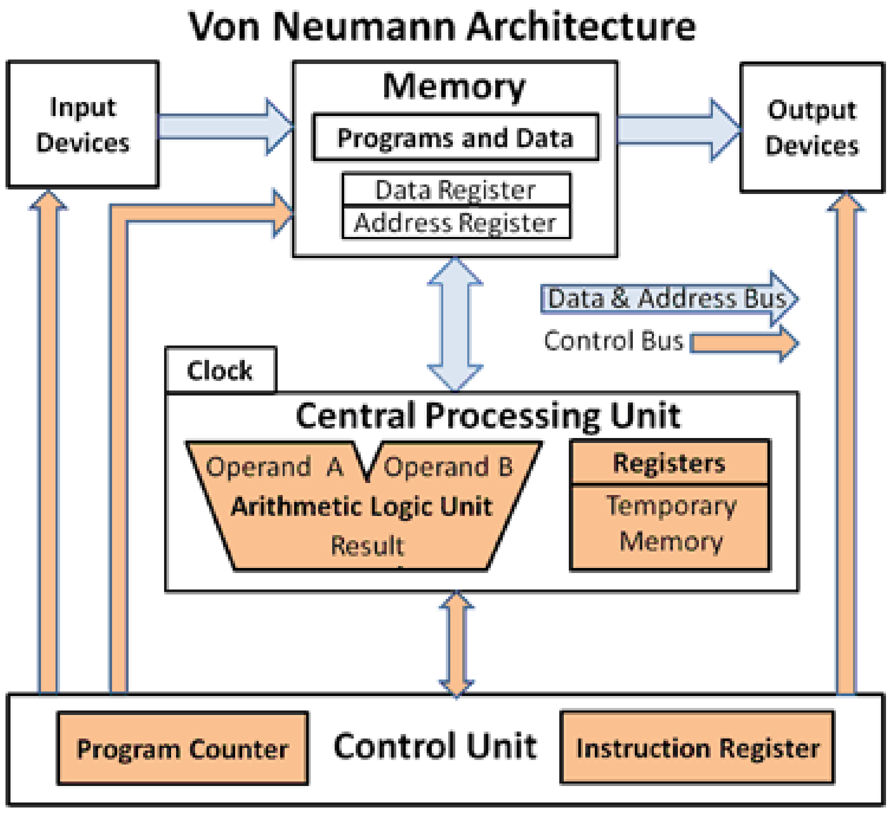
Introduction to Computer Architecture

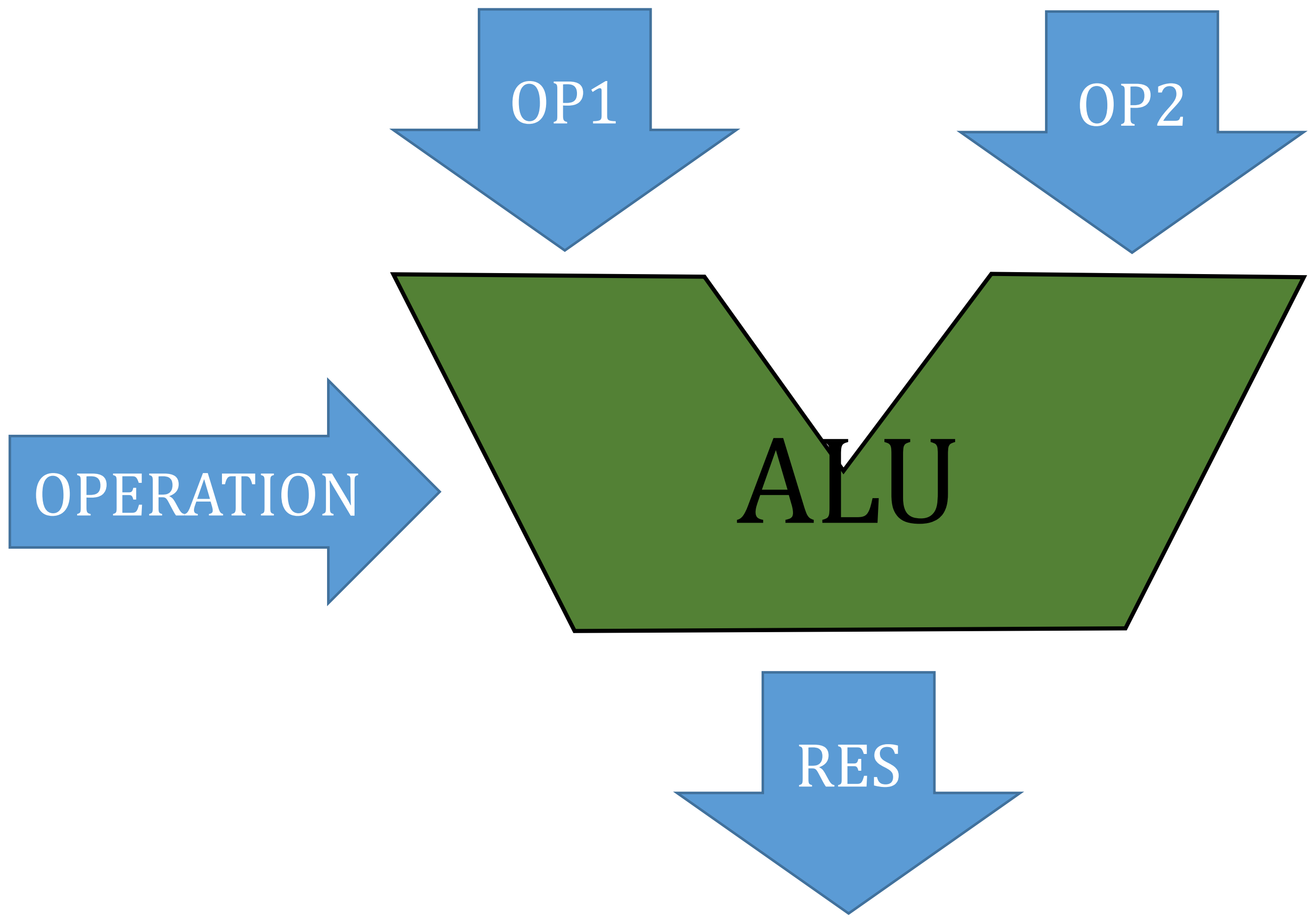
Chap 4.1



Suppose you built a computer…

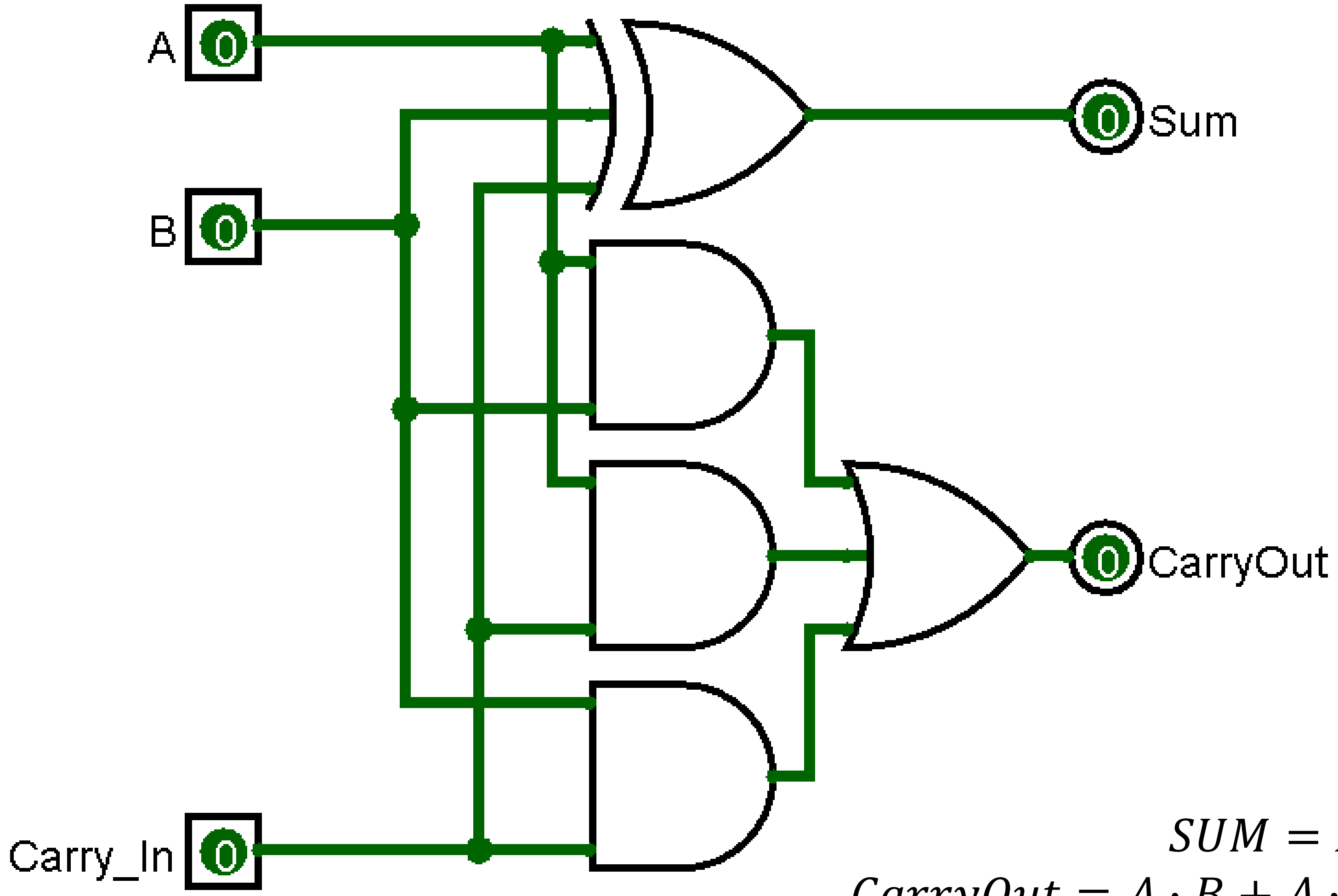
What Building Blocks would you use?

Arithmetic Logic Unit (ALU)



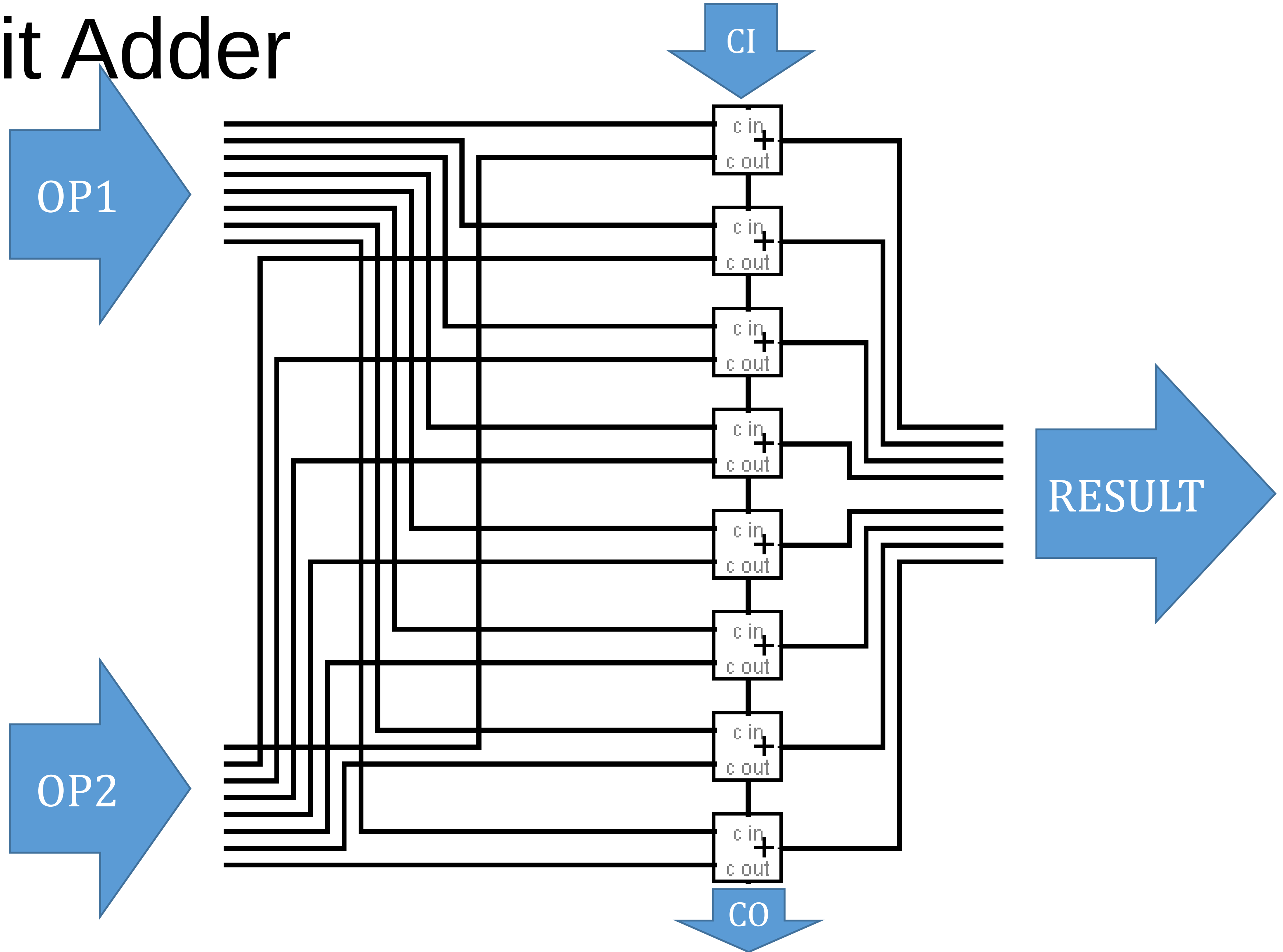
Full Adder

Chap 4.2

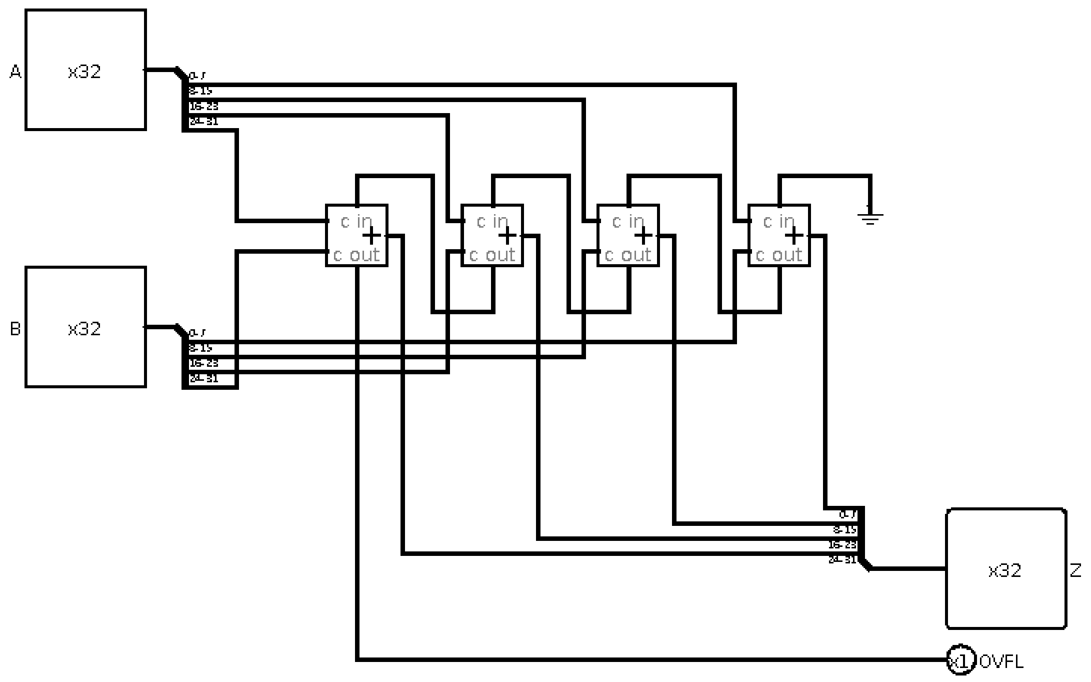


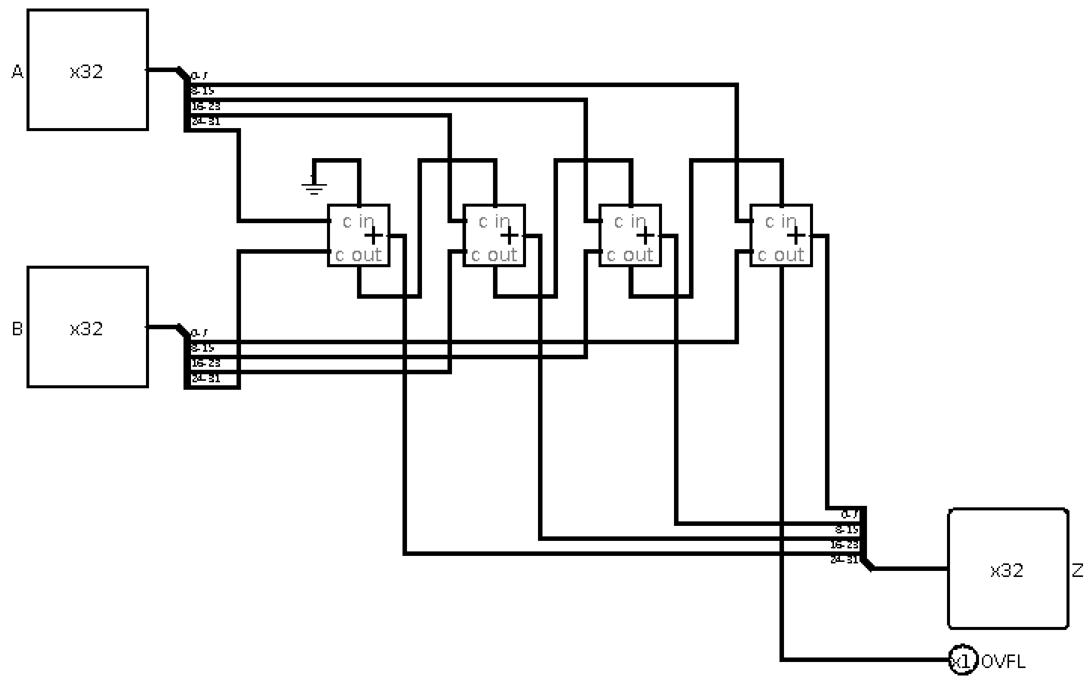
$$SUM = A \oplus B \oplus C$$
$$CarryOut = A \cdot B + A \cdot CarryIn + B \cdot CarryIn$$

Eight Bit Adder

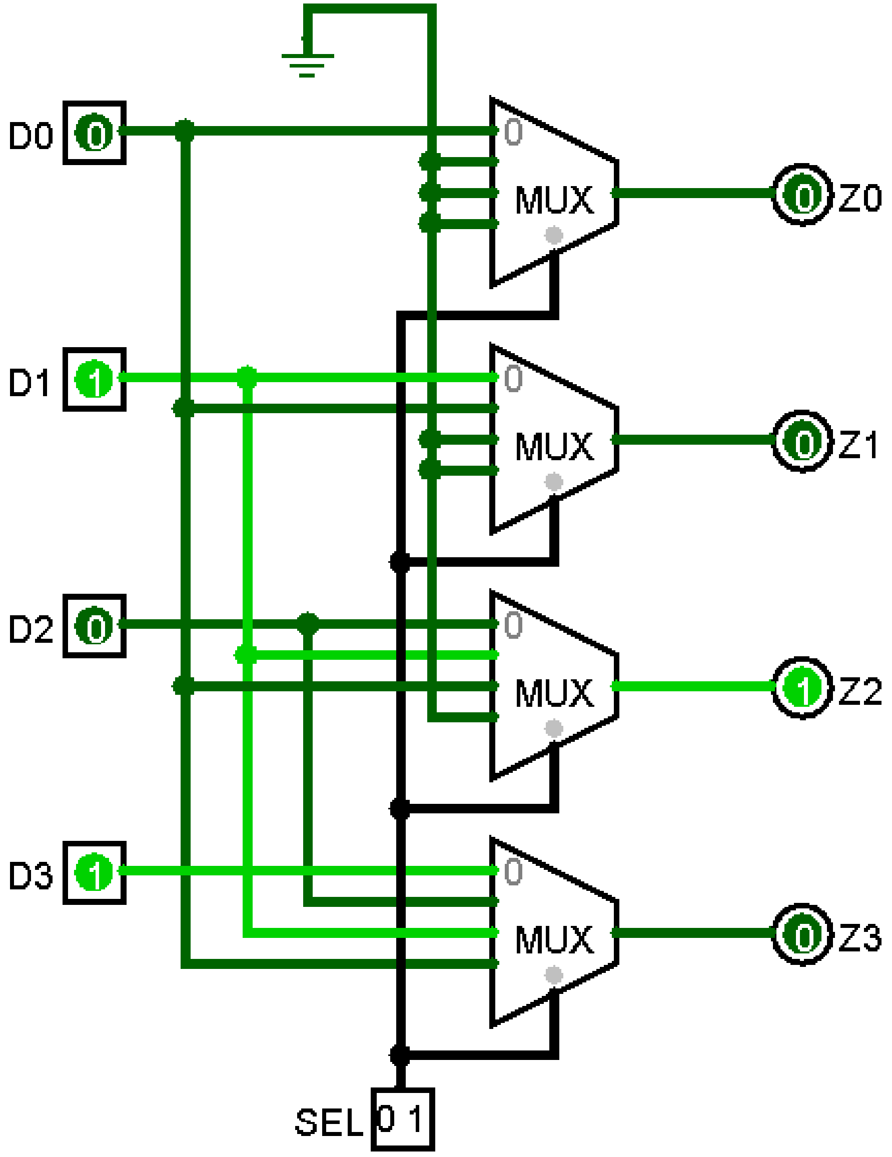


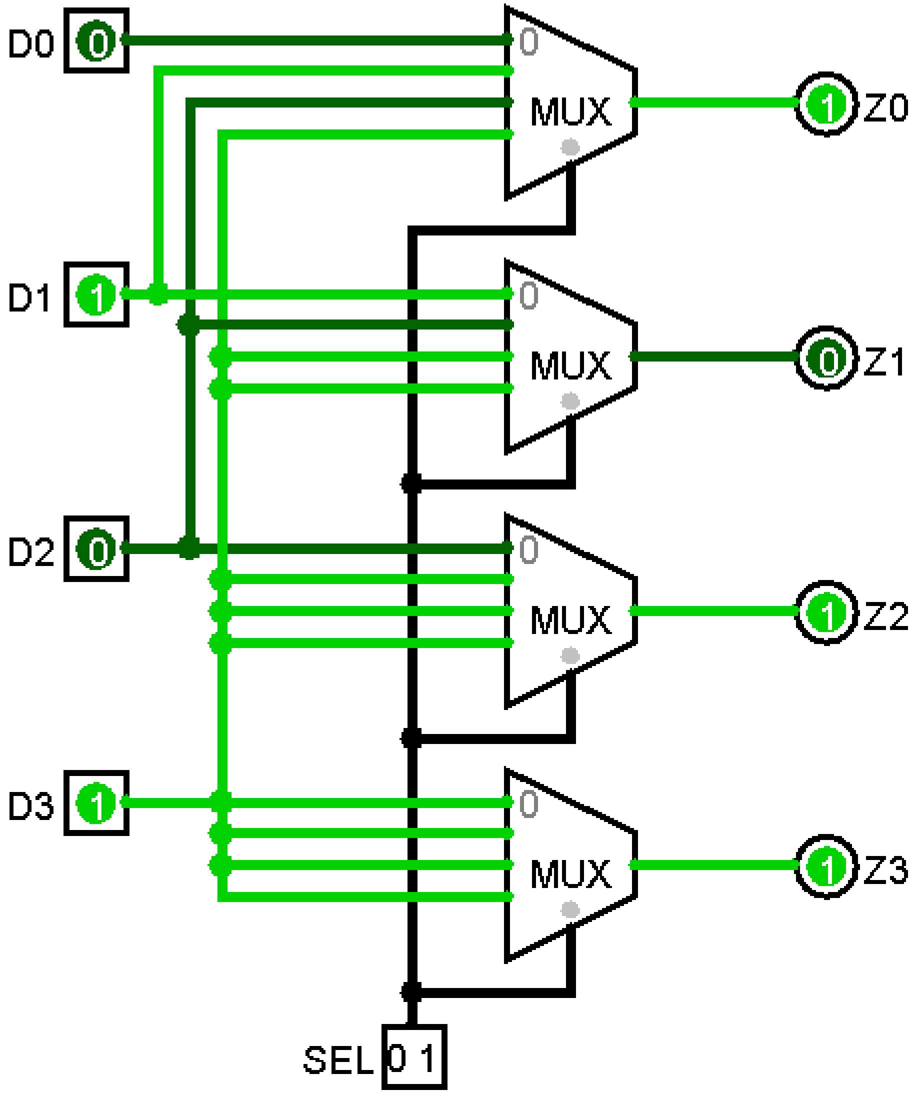
Big Endian vs. Little Endian Adder





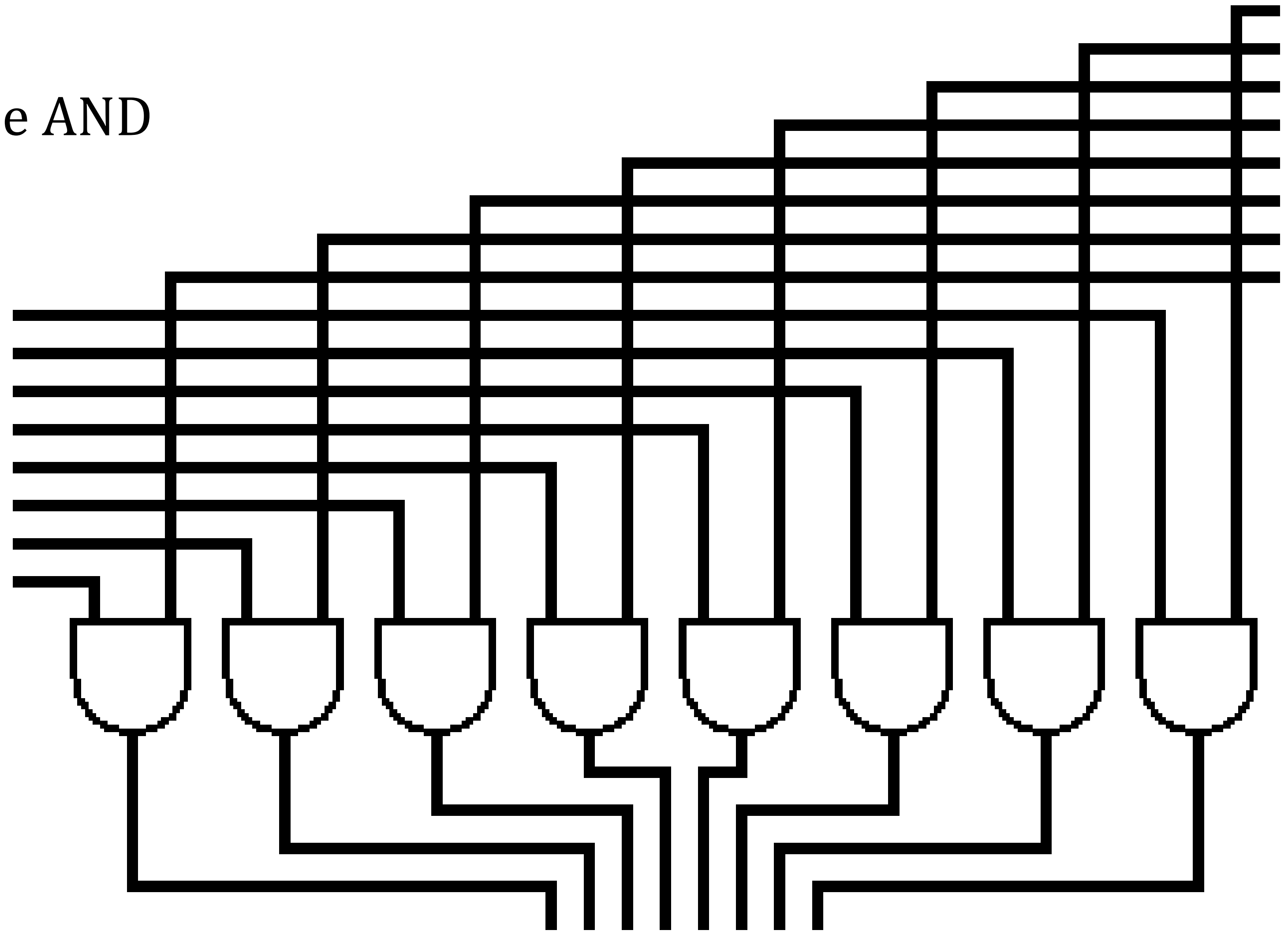
Shifters


D0
0
D1
1
D2
0
D3
1
MUX
0
Z0
Z1
Z2
Z3
SEL
0 1

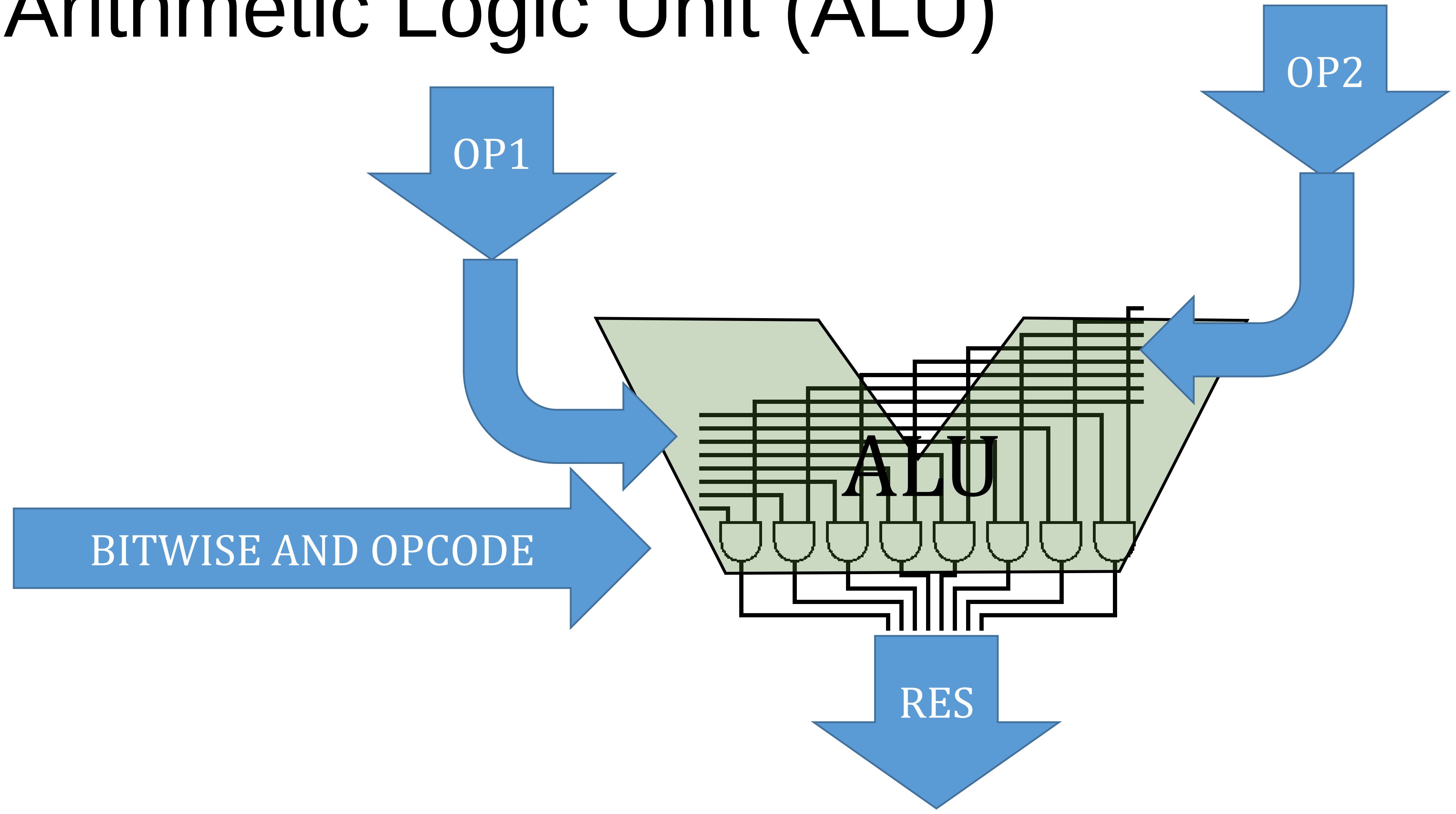

D0
0
D1
1
D2
0
D3
1
MUX
0
Z0
Z1
Z2
Z3
SEL
0 1

Hardware (Gate) Implementation

8 bit bitwise AND



Arithmetic Logic Unit (ALU)

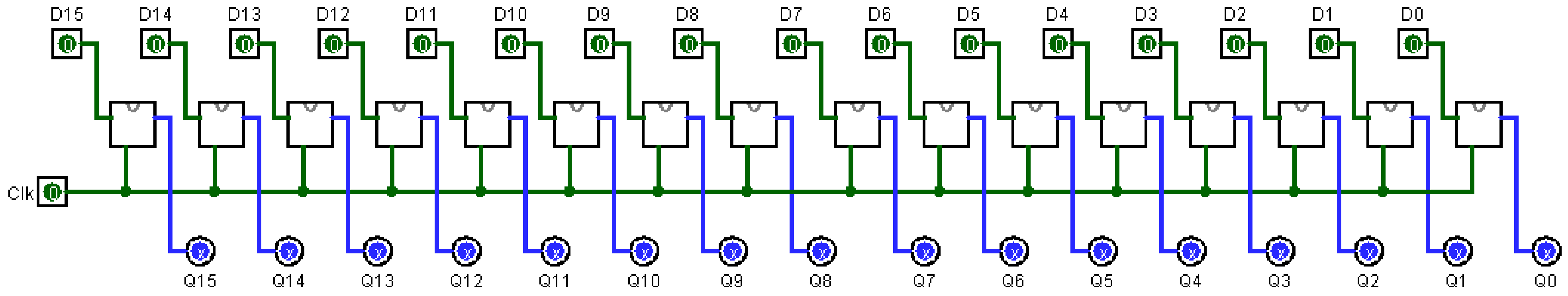


ALU Summary

- Reads 1 or 2 operands and produces a result
- Control to specify operation
  - Arithmetic (+.-,*,/)
  - Logic (bitwise or logical and/or/not/xor)
  - Shifting

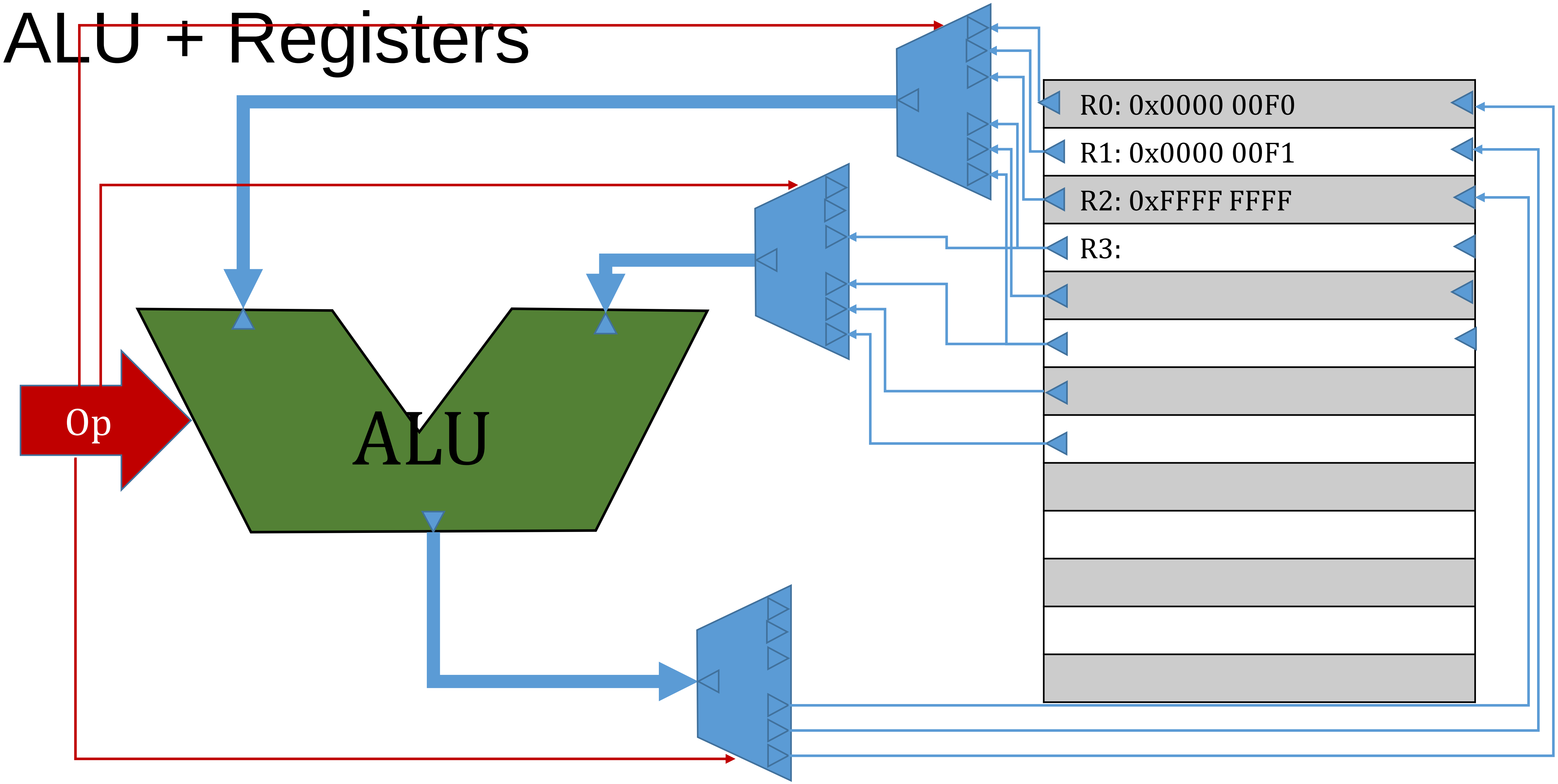
- But where does the data come from?

Integer Registers

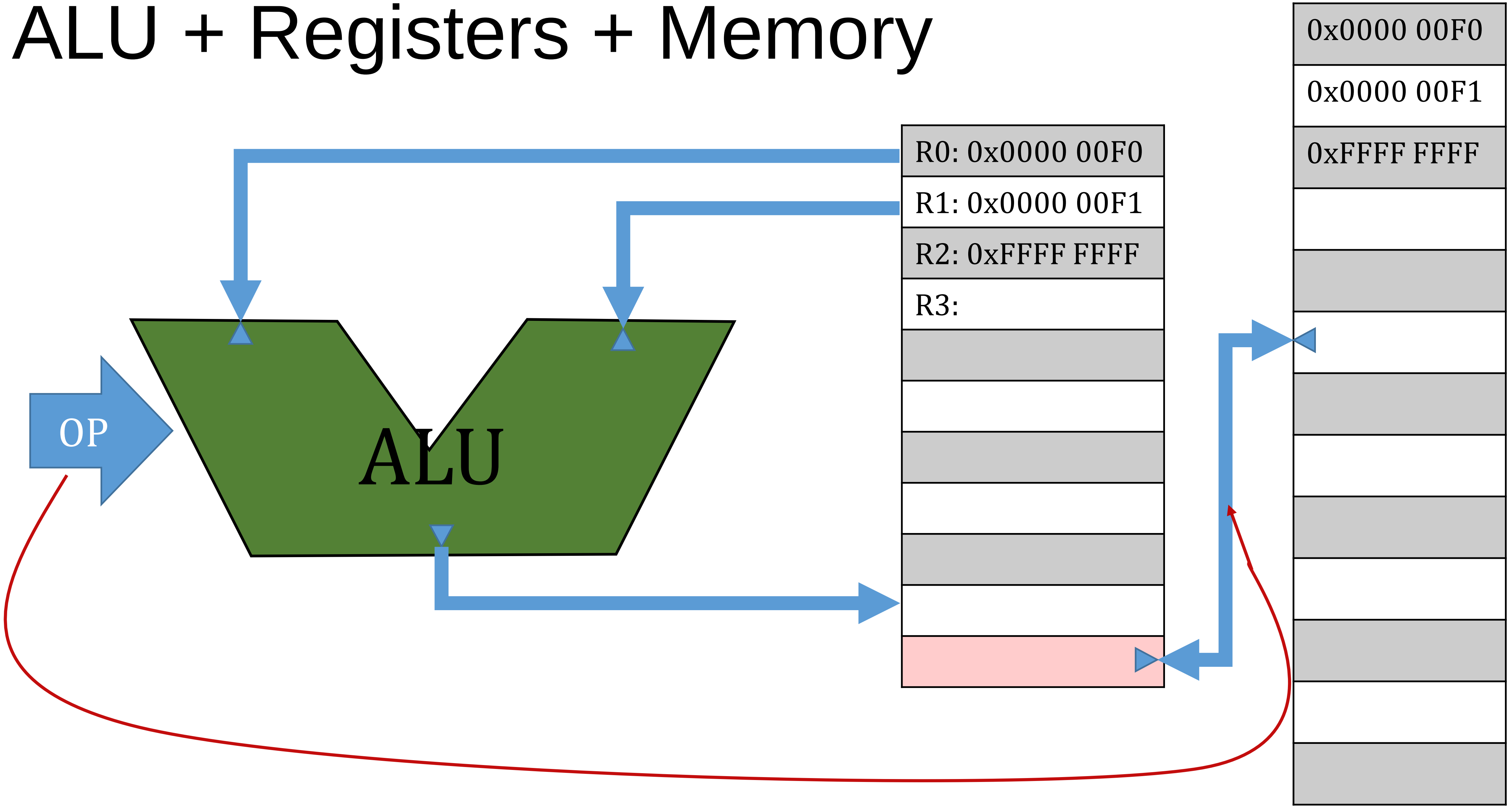


- Several 64 bit Registers built into the CPU
- Fast Read/Write (CPU Speed)
- No explicit data type!
- Values undefined (X) until set
- Use as operands for and result from ALU

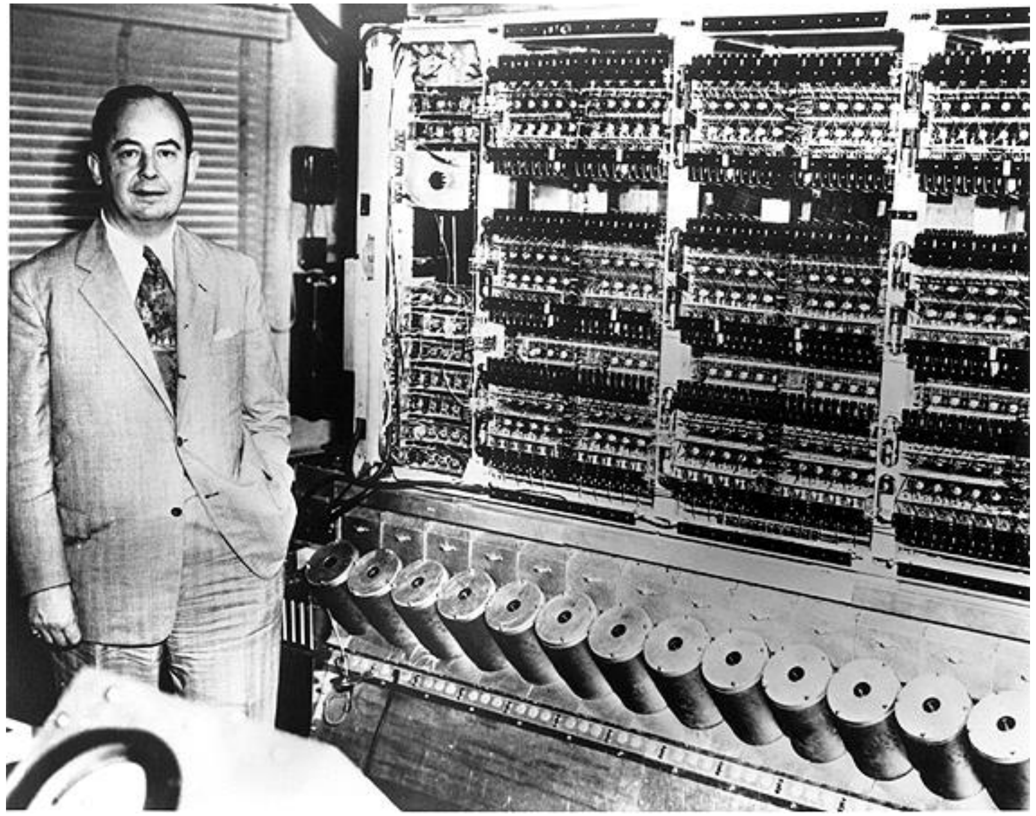
ALU + Registers



ALU + Registers + Memory

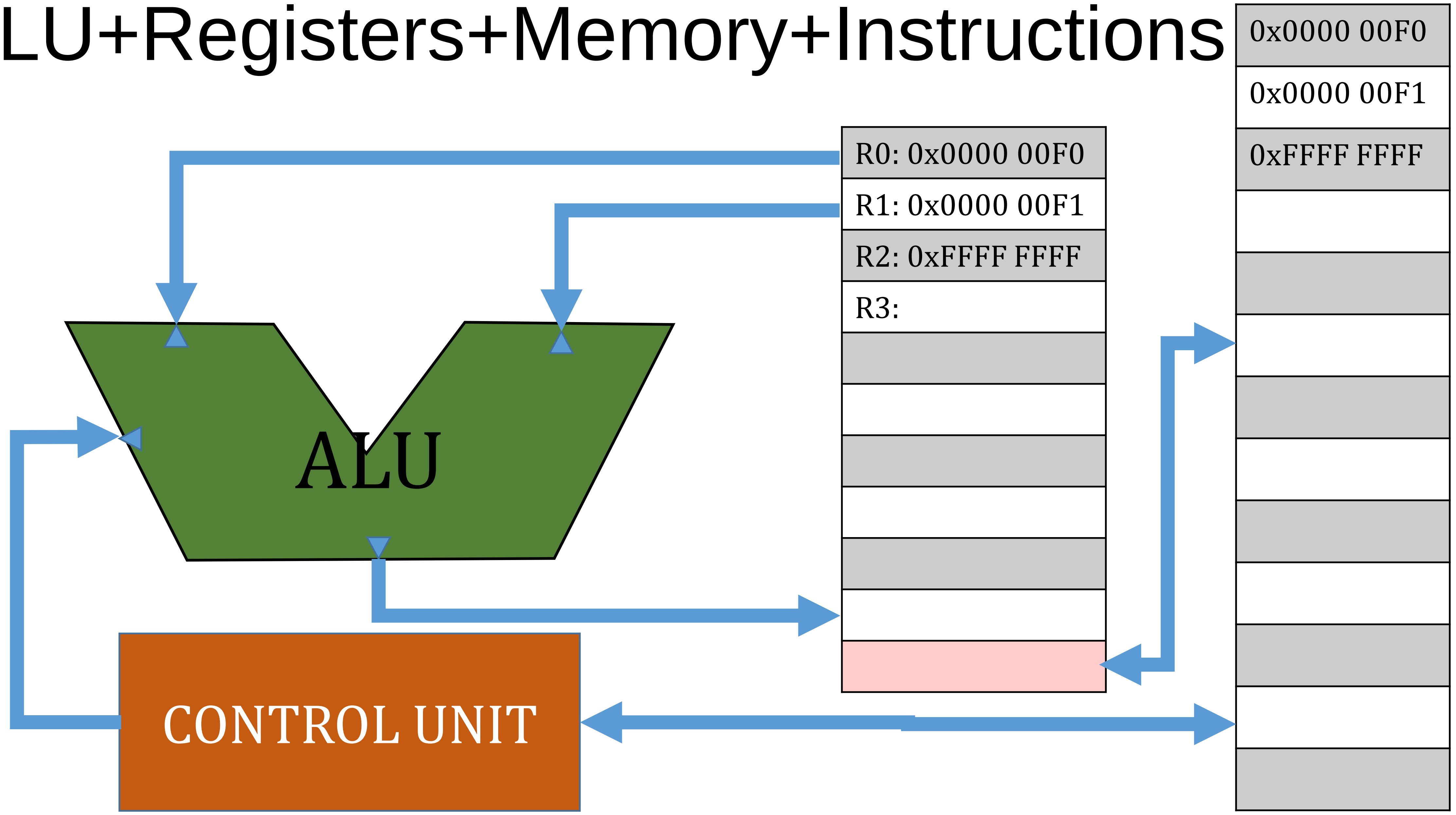


Computer vs. Adding Machine



What is the difference?

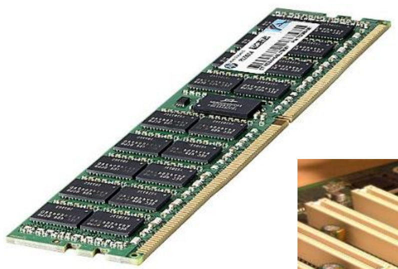
ALU+Registers+Memory+Instructions

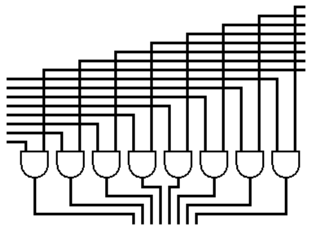


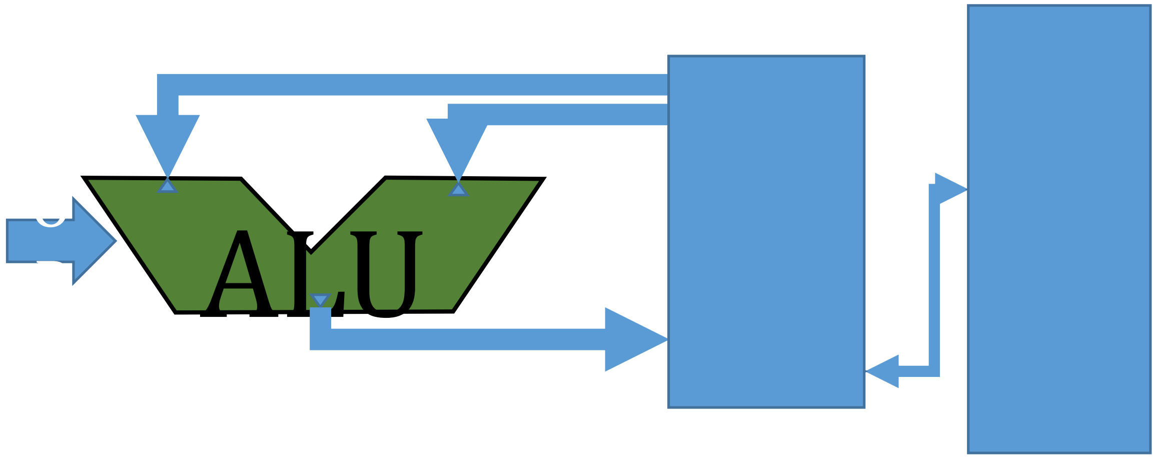
Hardware/Software Interface

```
int main(int argc, char **argv) {
    long n=atol(argv[1]);
    int i; printf("%ld = 0x",n);
    unsigned long mask=(1<<4)-1; // Four ones in the rightmost byte
    char * xd="0123456789ABCDEF";
    for(i=2*sizeof(n)-1;i>=0;i--) {
        int v=(n&(mask<<(i*4)))>>(i*4);
        printf("%c",xd[v]);
        if (0==i%4) printf(" ");
    }
    printf("\n"); return 0;
}
```

Instruction Set Architecture







Computer Architecture

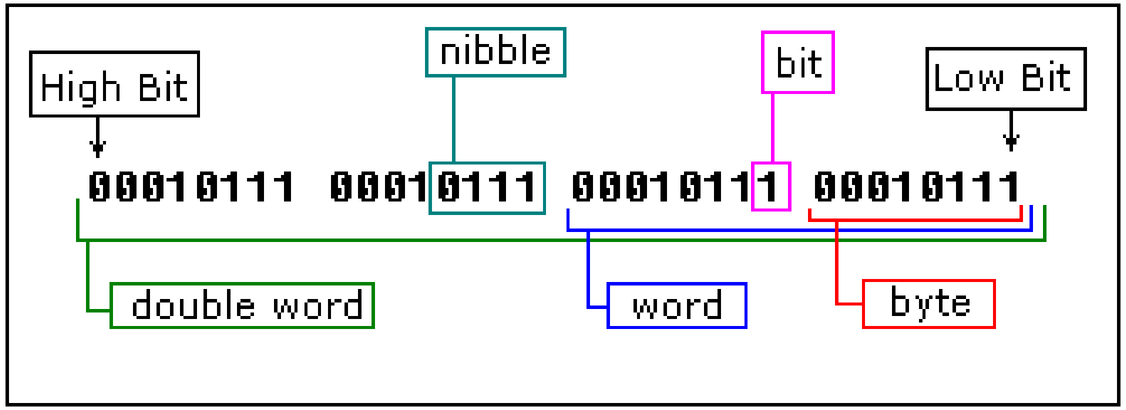
- Wikipedia “a set of rules and methods that describe the functionality, organization, and implementation of computer systems… the capabilities and programming model of a computer but not a particular implementation…”
- We will study the x86 architecture.
- Part of the architecture defines what type of data the hardware can operate on.

ISA Contents

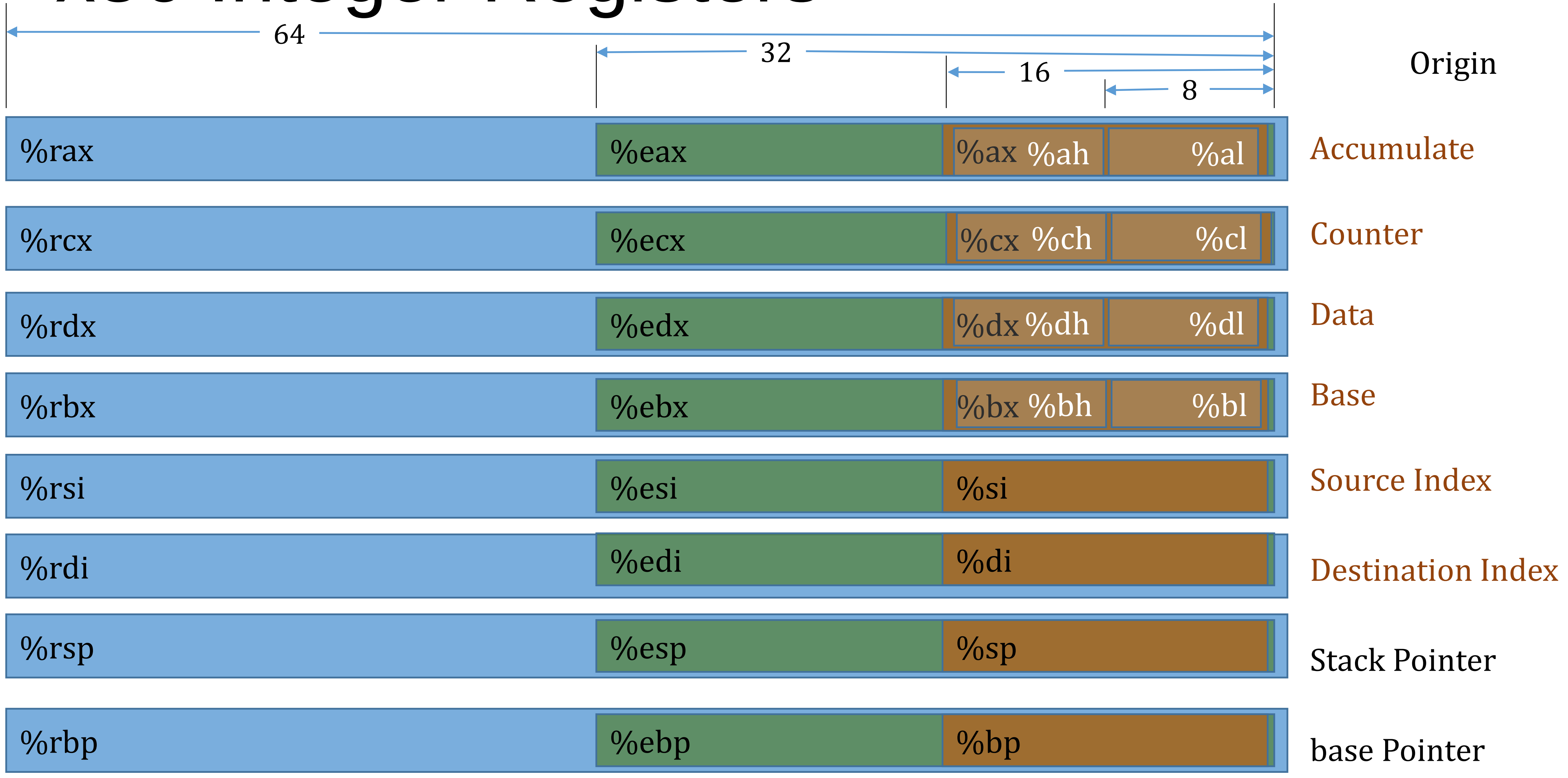
- The data types the instructions can work on
  - two’s complement binary, ascii character, unsigned binary, etc.
- The instructions the hardware recognizes
  - add, move, get, …
- The data the instructions can work on
  - Registers
  - Memory
- The external interfaces supported by the instructions
  - File I/O
  - Exception Handling and Interrupts

X86 Data Types

- Byte - 8 bit binary or ASCII character (char)
- Word - 16 bit binary (short)
- Double Word - 32 bit binary (int)
- Quad Word - 64 bit binary (long)
- 32, 64, or 128 bit floating point



x86 Integer Registers


64
32
16
8
Origin
%rax
%eax
%ax
%ah
%al
Accumulate
%rcx
%ecx
%cx
%ch
%cl
Counter
%rdx
%edx
%dx
%dh
%dl
Data
%rbx
%ebx
%bx
%bh
%bl
Base
%rsi
%esi
%si
Source Index
%rdi
%edi
%di
Destination Index
%rsp
%esp
%sp
Stack Pointer
%rbp
%ebp
%bp
base Pointer