ABC by Example and Experiments (Top-Down Approach)

More on Control Flow …

…and yes, there will be evil goto statements!

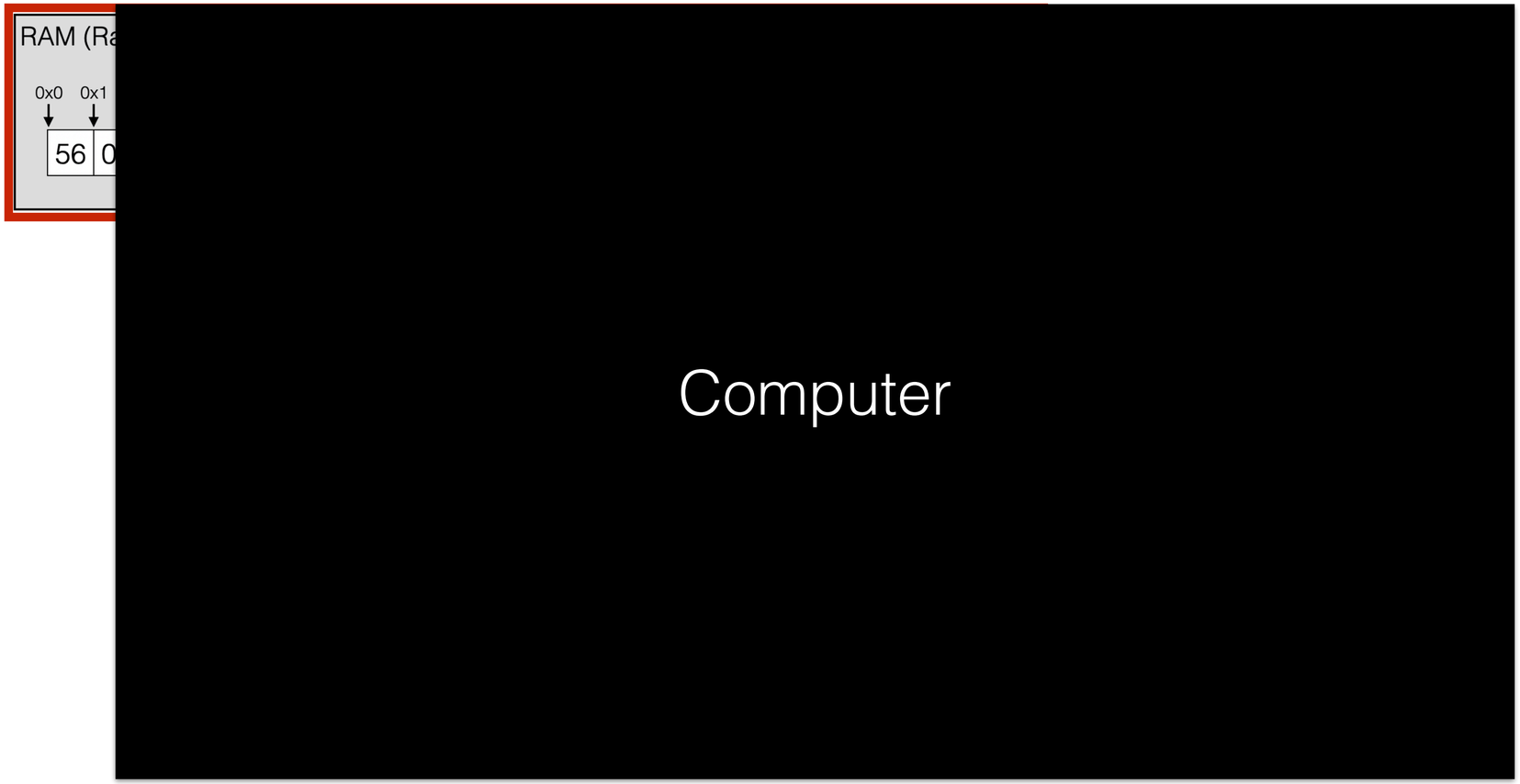
```
@ <stdio.hdr>

fn main()
{
    printf("A\n");
    printf("B\n");
    printf("C\n");
}
```

```
@ <stdio.hdr>

fn main()
{
    goto A;

label B:
    printf("B\n");
    goto C;


label A:
    printf("A\n");
    goto B;

label C:
    printf("C\n");
}
```

```
@ <stdio.hdr>

fn main()
{
    goto A;

label B:
    printf("B\n");
    goto C;


label A:
    printf("A\n");
    goto B;

label C:
    printf("C\n");
}
```

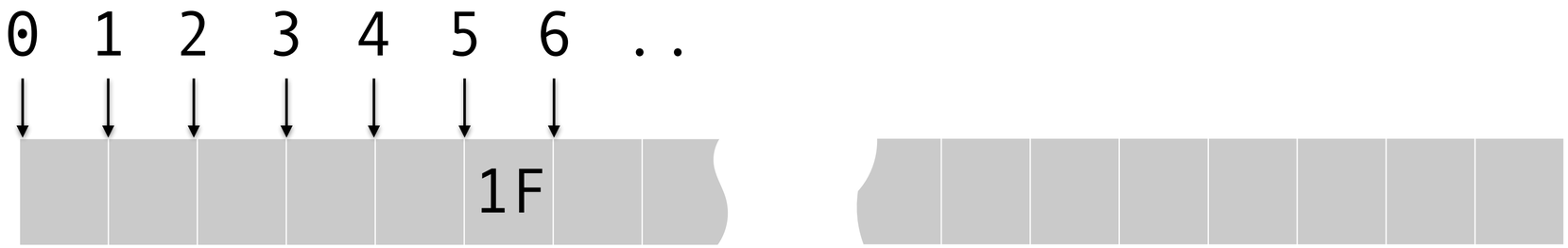
```
@ <stdio.hdr>

fn main()
{
    putchar('A');
label B:
    putchar('B');
    putchar('C');
    putchar('D');
    putchar('E');
    putchar('F');
    goto B;
    putchar('G');
}
```

Some Technical Background

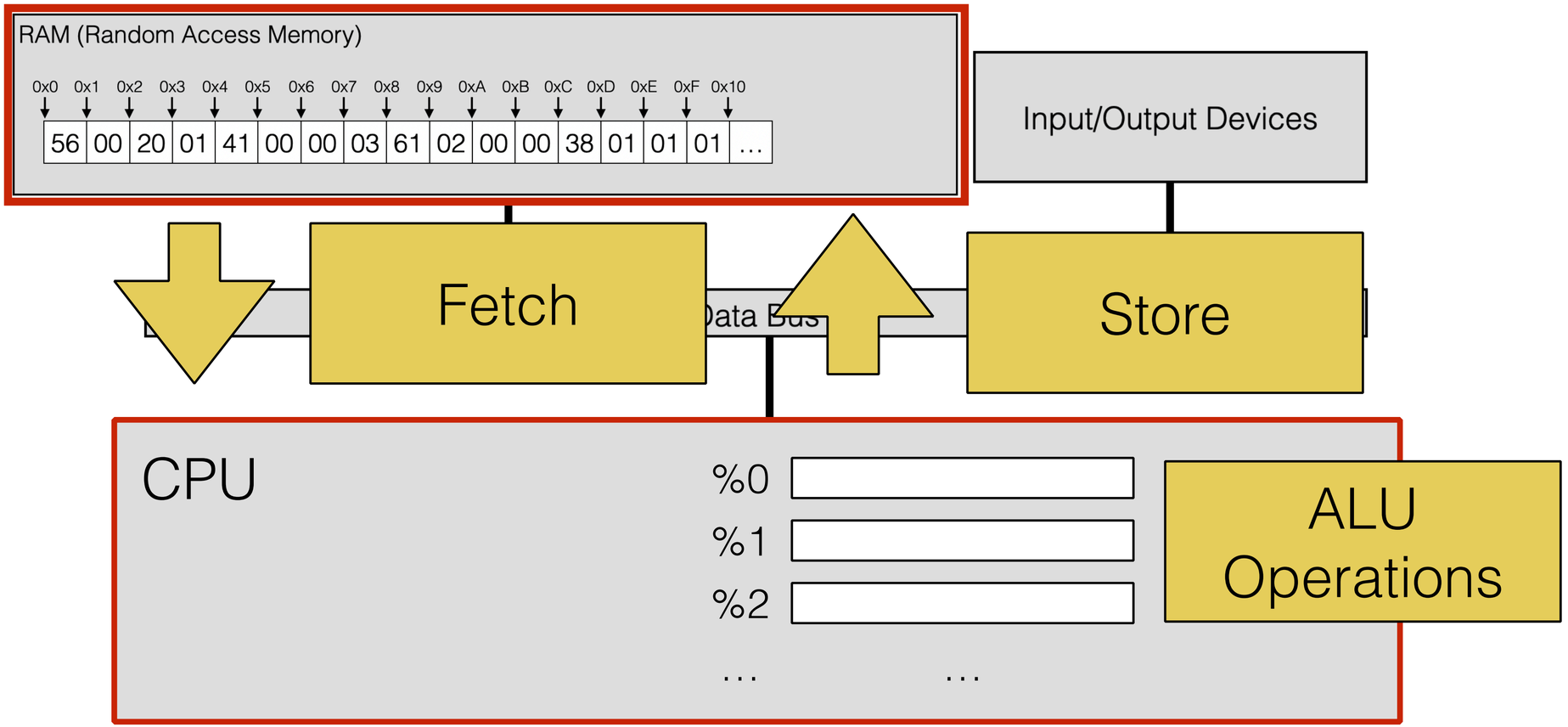


RAM (Random Access Memory)

- Array of memory cells
- Size of a cell is called **Byte** (at least 8 bits)
- Every cell has an address (first address is 0)

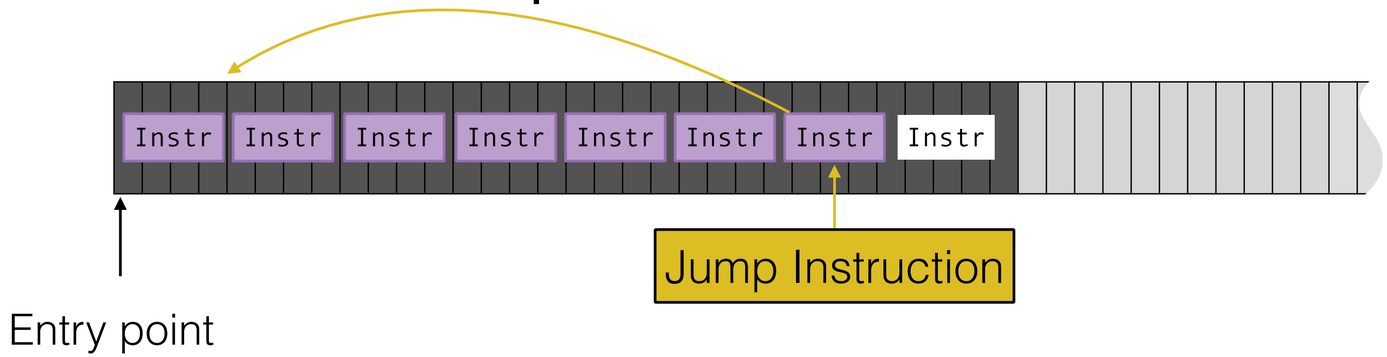


Some Technical Background



64-bit CPU registers %0, %1, %2, …

Sequential execution



- Execution begins with the first instruction and
- Continues with the “next” instruction
  - Usually the instruction that comes next in memory
  - Jump instructions can influence what to do next.

```
@ <stdio.hdr>

fn main()
{
    local n: int = 5;
    local res: int = 1;

    printf("%d! = ", n);

    while (n >= 2) {
        res *= n;
        --n;
    }

    printf("%d\n", res);

}
```

```
@ <stdio.hdr>

fn main()
{
    local n: int = 5;
    local res: int = 1;

    printf("%d! = ", n);
label loop:
    if (n < 2) { goto done; }
    res *= n;
    --n;
    goto loop;

label done:
    printf("%d\n", res);
}
```