Introduction to Computer Science

Lecture 3: Negative Numbers
How computers execute programs

Iain Styles
I.B.Styles@cs.bham.ac.uk

Recap

- Last time:
  - Fixed point numbers
  - Arithmetic in binary
- This time:
  - Negative numbers
  - Floating point
  - Introduction to how computers execute programs

Negative numbers

- We need to be able to represent negative numbers purely in binary
- In decimal representation, we have ten symbols plus the “minus” sign
- In binary, only two symbols
- So how do we denote the “sign” of a number?

Try something obvious

- Add a “sign” bit – assume on the left
- Then +5 = **0**101 and -5 = **1**101
- We know that +5 + -5 = 0

```
           0 1 0 1
     +     1 1 0 1
(Carry     1   1    )
         1 0 0 1 0
```

- We need to be a bit smarter about this...

An observation about overflow

- Let's investigate overflow a bit more closely

```
          0 1 1 1
    +     1 1 1 1
(Carry    1 1 1    )
      (1) 0 1 1 0
```

- Interesting...try something else

```
          1 0 1 0
    +     1 1 1 1
(Carry    1 1      )
      (1) 1 0 0 1
```

An observation about overflow

- And again

```
         0 1 1 1
     +   1 1 1 0
(Carry   1 1 1   )
     (1) 0 1 0 1
```

- And one more

```
         1 0 1 0
     +   1 1 0 1
(Carry   1 1     )
     (1) 0 1 1 1
```

Two's Complement

- Addition of a “large” number plus overflow looks like subtraction

  +111...111 $\rightarrow$ -1

  +111...110 $\rightarrow$ -2

  +111...101 $\rightarrow$ -3

- This leads us to the **two's complement (2C)** representation

Two's Complement Arithmetic

- In 2C the convention is:
  - All positive numbers start with 0
  - All negative numbers start with 1
  - Negation is achieved by:
    - Flipping all the bits
    - Adding 1 to the least significant (right-most)
- 011010 (positive) → 100110 (negative)

```
            0 1 1 0 1 0
        +   1 0 0 1 1 0
(Carry 1 1 1 1 1       )
        (1) 0 0 0 0 0 0
```

Have a go

- 2C is used to implement subtraction
  - Easier to calculate 2C and add than to subtract
- 0001 - 0110?
- 1110 – 1001?

Range of 2C

- Given N bits, what is the largest value of a 2C number?
  - Recap: 111...111 = 2^N-1
  - Then 011...111 = 2^{N-1}-1
- What is the smallest (most negative) value?
  - 100..000 = (-)100...000 = -2^{N-1}
- Example: 8 bits
  - Maximum: 2^7-1 = 127
  - Minimum: -2^7 = -128

Fixed Point Arithmetic

- Everything is the same as for whole numbers
- Example: 01001.010 – 00010.100
- Take 2C and add:

```
          0 1 0 0 1 0 1 0
       +  1 1 1 0 1 1 0 0
(Carry  1         1            )
      (1) 0 0 1 1 0 1 1 0
```

- 00110.101 – 10110.010?

Floating Point

- Fixed point trades accuracy for range
- A possible alternative:

$$V = M \times 2^{E}$$

  - Why? Powers of 2 are easy to compute in binary!
  - M is the *mantissa:* 2C fixed point, one integer bit
  - E is the *exponent*: 2C integer
  - So 0101 1001 = $0.101 \times 2^{0110} = 0.625 \times 64 = 40$
  - 1101 1110 = ?

Summary

- Binary arithmetic is like decimal arithmetic
  - But we are not practised so find it hard
- Negative numbers are tricky things
  - But we can use a few of our own tricks – 2C.
- Floating point is an alternative, but is very unnatural for us

Next time, we will begin to study how computer are organised, and how they execute programs

How Computers Execute Programs

Here is a simple computer program

```
a = input('Enter a number');
if a<1 exit('Invalid input');
b=1;
for(i=1;i<=a;i++) {
   b=b*i;
}
print (a,'! = ',b);
```

Requirements of a Computer

- Process instructions correctly and in the correct order
- Access and modify pieces of data in accordance with the program’s instructions
- Take decisions according to the results of the computations
- Be able to repeat parts of itself
- Send the results of the the computations to some external device or store them for further use

Computers must be able to

Load/store instructions and data to/from the outside world

Store instruction and data locally

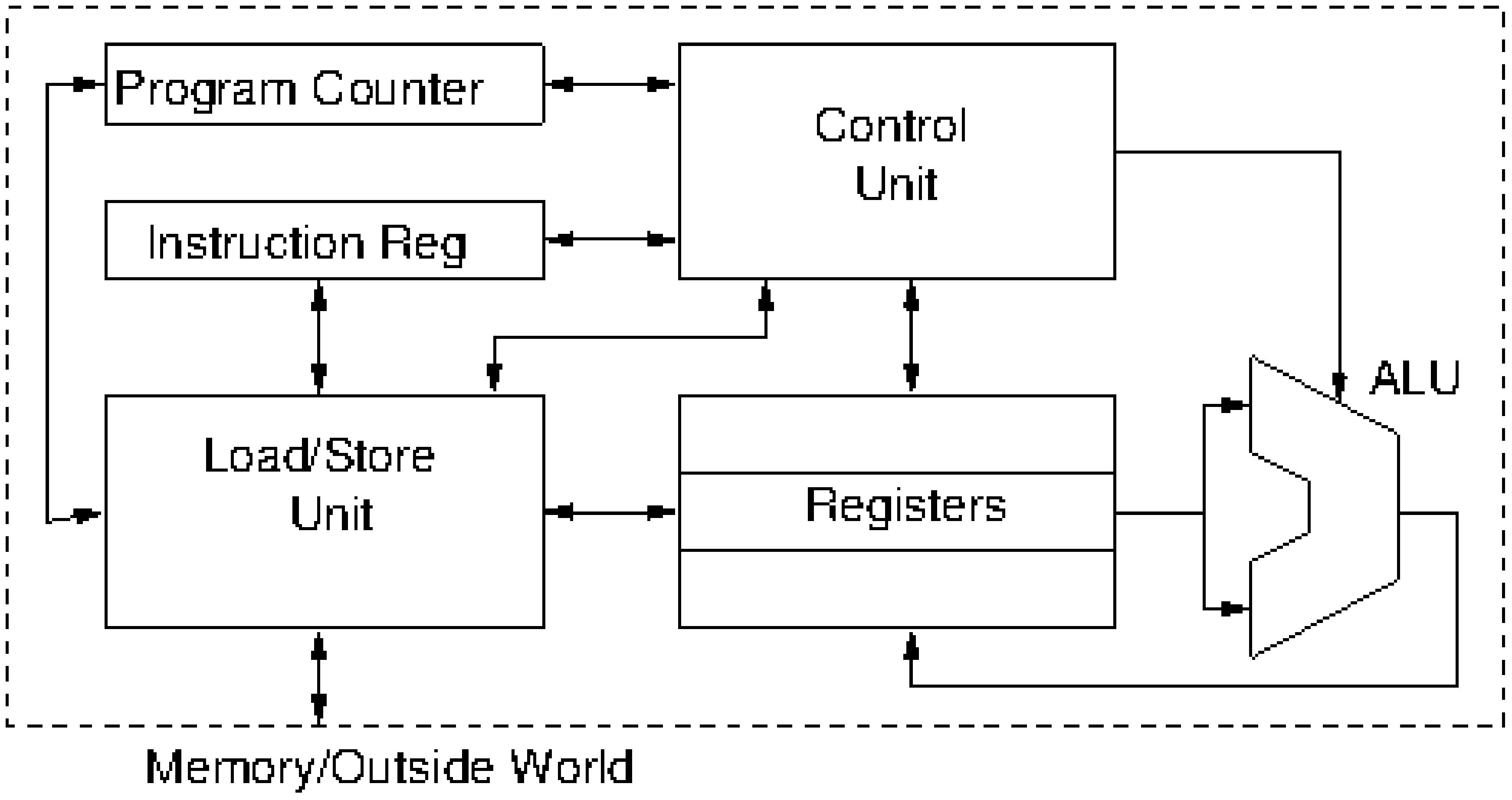
Interpret the instructions to do the necessary computations

Send the results to external devices

Take appropriate decisions and control what’s happening

Keep track of program execution

The von Neumann Model



Summary

- We've considered the requirements of a simple computer program
- And looked at an outline design that we (I?) claim could execute it.
- Next time, we will look at this design in some more detail by studying a real example that implements it.