Assembly Programming

Nicholas Miehlbradt

October 21, 2022

What is assembly?

- Computers read raw bytes from memory. The pattern of bytes correspond to instructions.
- Assembly is a more human readable version. There is still a 1-to-1 correspondence with instructions.
- ISAs will define the assembly syntax and the translation

$$\text{add r3, r4, r2} \leftrightarrow \text{0x8342}$$

$$\text{int x} = 5 * \text{y} + 3 \leftrightarrow ???$$

Lets write a program

We want a program which reads two numbers from memory, adds them together and stores the result back to another location in memory.

int a = 5
int b = 10
int c = a + b

Assemblers

Hand assembling code is tedious and error prone. Luckily these are things computers excel at. Programs which convert assembly to machine code are called assemblers.

Lets write an if statement

Compare two numbers. If they are equal store 1, else store 0.

```
if (a == b)
c = 1
else
c = 0
```

Labels

We can use a label to 'bookmark' a section of code. The assembler will make note of what addresses labels correspond to and substitute in the correct numbers.

```
start:
add r1, r2, r3
jnz start
```

Fibonacci Numbers

Write a program that computes Fibonacci numbers. Can you make the program stop when it reaches a specific number?

Functions

To be able to use functions we need to both jump to a new instruction and remember where we came from.

The call instruction saves the address of the next instruction in the link register (lr).

The ret instruction copies the value from the link register to the program counter.

Calling Conventions

We need a way to pass information between a function and it's caller. We don't have unlimited variables so we need to nominate specific places that the caller and callee will agree on.

Registers are shared so we need to agree on who will save registers if both caller and callee want to use them.

These agreements are called the calling convention. There is nothing forcing the programmer to follow it but...

Lets write a function

Multiplication is useful but we don't have a multiplication instruction. Can you write a multiply function and then use it in multiple places?

The Stack

What happens when we want to call a function inside another function? We need to save our link register before calling it.

We need a region of memory that we can grow and shrink as we call and return from functions.

Note: there is nothing enforcing the correct use of the stack, it is up to the programmer to use the correct instructions!

Recursive functions

Can you write a function which calculates a factorial using recursion?

Higher Level Languages

As you've probably realised, programming in assembly is very tedious.

We have come up with higher level languages which add constraints but also hide the complexity. Let computers do the bookkeeping so we can focus on the logic.