

cse141 Project: Design Your Own ISA

Overview

- You will design an Instruction Set Architecture
 - Define your instructions and your calling convention
 - Produce an assembler
 - Produce a simulator.
 - Demo your tools for Hung Wei.
- Due October 20th
- Start today

Your ISA

- Unusual constraints keep things interesting
 - 34 bit addresses/data
 - 17 bit instructions
 - Mimicking MIPS will not work.
- Needs to be “general-purpose”
 - In particular, it needs to run two programs given on the website.
 - It can be more general.
- Be creative!!!
 - Do something crazy! I like crazy! I will reward crazy!

Program I: Fibonacci

```
int fib(int n)
{
    if (n < 0)
        return 0x3DEADBEEF;
    else if (n <= 2)
        return 1;
    else if (n == 29)
        return 514229;
    else if (n == 30)
        return 832040;
    else if (n == 48)
        return 4807526976;
    else if (n == 49)
        return 7778742049;
    else return fib(n-1) + fib(n-2);
}
```

Program II: SuperGarbage

```
struct inst {
    int op;
    int srcA;
    int srcB;
    int dest;
};

int SuperGarbage(int pc, int *mem)
{
    while(1)
    {
        struct inst *instruction = &(mem[pc]);
        int op = instruction->op;
        int srcA = instruction->srcA;
        int srcB = instruction->srcB;
        int dest = instruction->dest;
        pc = pc + 4;

        switch(op) {
            case 0: mem[dest] = mem[srcA] - mem[srcB]; break;
            case 1: mem[dest] = mem[srcA] >> 1; break;
            case 2: mem[dest] = ~(mem[srcA] | mem[srcB]); break;
            case 3: temp = mem[srcB];
                    mem[dest] = mem[mem[srcA]];
                    mem[mem[srcA]] = temp; break;
            case 4: in mem[dest], mem[srcA]; break; // in mem, channel #
            case 5: out mem[srcA], mem[srcB]; break; // out data, channel#
            case 6:
                    mem[dest] = pc;
                    if (mem[srcA] < 0)
                    {
                        pc = mem[srcB];
                    }
                    break;
            case 7: return pc;
        }
    }
}
```

- All your ISAs will be different.
- How do we test them?
- We use a virtual machine!!!
 - Very simple (but complete!) instruction set.
- We provide the SuperGarbage assembler

The Tools: Assembler

- Your assembler will generate binaries for your simulator (and your eventual processor in I4IL)
- We provide a framework, that makes building the assembler pretty easy.
 - Don't be afraid to hack it, if you need some additional functionality.
 - Talk to Hung Wei
- Features your assembler will support
 - Code and data sections
 - Pseudo instructions
 - Labels
 -

Tools: The Simulator

- This is an ISA Simulator, or an interpreter -- Not a hardware simulator.
- It is only concerned with the big-A effects of the program its running.
 - The IO
 - The architectural state -- memory, registers, PC, etc.
- We provide a framework for this too.
 - It's basically a while loop and a case statement.
 - Questions? Talk to Hung Wei