COMS12200 Introduction to Computer Architecture

Simon Hollis (simon@cs.bris.ac.uk)

Machine types and paradigms

COUNTER MACHINES

Counter machines

- Counter machines are one of the simplest possible machines to create.
- They are register machines
  - Register length is unbounded
  - #registers: 1→∞
- The instruction set is very limited

Unbounded registers

- Registers are unbounded
  - Allows them to store an infinite amount of information
    - Provided they use a Gödel encoding
  - Unrealistic for a real machine, but very nice for theoretical applications.
- Counter machines with finite length registers and finite number of registers exist and are very easy to implement

Counter machines can be Turing powerful

- A counter machine with at least two unbounded registers is Turing powerful.
  - Dual-stack analogy
  - Gödel encoding + scratchpad analogy

Counter instruction set

Here's the cool thing: counter machines can perform any computation…

…however, they need more than three different instructions to do this.

Most popularly, these three instructions:

1. INC *r*   (increment register *r*)
2. DEC *r* (decrement register *r*)
3. JZ *r* (jump if *r* = zero)

- *Also see* JEQ r_i, r_j (jump if $r_i = r_j$)

Counter machines memory

- Counter machines use the Harvard memory paradigm.
  - The instructions are therefore separated out.
  - There is an implicit PC defined
    - But only updatable via jumps
  - Data values are only stored in the defined registers.

All other operations are synthesisable

The super-cool thing about the counter machine definition is that all other operations we might want are synthesisable from the three instructions we have (*INC*, *DEC*, *JZ*).

Examples:

- *COPY* r_i, r_j
- *ADD* r_i, r_j

Simulation efficiency

- In general, the efficiency of simulation of a Turing Machine programme is O($1/2^N$) [i.e. terrible!].
- However, the inclusion of some more complex instructions can improve this run-time dramatically.
  - Note that many alternative register machine implementations are much more efficient than a Turing Machine.

Counter machine implementation

? How can we implement a counter machine?

? How complex is a counter machine implementation?

? Would we ever want to implement a counter machine?

Summary

- Counter machines are interesting minimal examples of register machines
- They require surprisingly few instructions to operate powerfully
- We need only three instructions to do anything!
- The run-time efficiency depends on the instructions available.
- They are implementable in finite state form

Appendix:
Example templates

Counter instruction set

Instruction Code	Argument	Meaning
1	*r*	INCrement *r*
2	*r*	DECrement *r*
3	*r*, *i*	Branch to instruction *i* if *r* == *0*
4	-	Halt processing

Counter machine example

Register 1	Register 2	Register 3	Register 4

Instruction memory

Address	Content

Data Memory

Address	Content