COMS30046 Advanced Computer Architecture

(Interim submission)

Archie Preston

ISA

The ISA implemented by the simulator is a custom specification, created as a very minimal ISA for proof-of-concept functionality. Instructions take multiple types of arguments:
Registers, denoted `rn` (e.g. `r1`, `r2`); Immediates, denoted `#n` (e.g. `#1`, `#64`); and Labels, denoted by any other string.

For describing the ISA, register aruments will be denoted r_n, immediate argumenst i_n, and operands arguments (either register or immediate) o_n. Locations, which can be registers, immediates, or labels (which get precompiled into immediates) are denoted l_n.

ISA

Instruction	arg 1	arg 2	arg 3	Functionality	Instruction	arg 1	arg 2	arg 3	Functionality
`add`	r_1	o_1	o_2	$r_1 \leftarrow o_1 + o_2$	`st`	o_1	o_2	_	$mem[o_1] \leftarrow o_2$
`sub`	r_1	o_1	o_2	$r_1 \leftarrow o_1 - o_2$	`b`	l_1	_	_	$\text{pc} \leftarrow l_1$
`mul`	r_1	o_1	o_2	$r_1 \leftarrow o_1 \times o_2$	`j`	o_1	_	_	$\text{pc} \leftarrow \text{pc} + o_1$
`not`	r_1	o_1	_	$r_1 \leftarrow \neg o_1$	`bilz`	r_1	l_1	_	$\begin{cases} \text{pc} \leftarrow l_1 & \text{if } r_1 < 0 \\ \text{pc} \leftarrow \text{pc} & \text{otherwise} \end{cases}$
`and`	r_1	o_1	o_2	$r_1 \leftarrow o_1 \wedge o_2$	`jilz`	r_1	o_1	_	$\begin{cases} \text{pc} \leftarrow \text{pc} + o_1 & \text{if } r_1 < 0 \\ \text{pc} \leftarrow \text{pc} & \text{otherwise} \end{cases}$

Current project status

- Functioning parser (does not yet comprehend text labels)
- Functioning basic interpreter - processor can return valid halting state of an arbitrary input program

TODO:

To catch up

- Implement correct cycle counting
- Create concrete systems to dispatch states to
    - Use these systems to construct pipelined execution
- Write more benchmarks

I am aware I'm very behind here, sorry!

TODO:

Going forward

- Create a 2-bit branch predictor
- Implement Out-of-Order execution
- Finish writing bubble sort in the ISA
- Write 2 other programs in the ISA (likely matrix multiplication and a collatz calculator)