Lecture A6: Building an X-TOY

The X-TOY Machine ISA

- 256 16-bit words of memory.
- 16 16-bit registers.
- 1 8-bit program counter.
- 16 instructions types.

What we’ve done.
- Software implementation of fetch-execute cycle.
  X-TOY simulator

Our goal today.
- Hardware implementation of fetch-execute cycle.
  X-TOY computer

Designing a Processor

How to build a microprocessor?

- Develop instruction set architecture (ISA).
  - 16-bit words, 16 TOY machine instructions

- Determine major components.
  - memory, registers, ALU, program counter

- Determine datapath requirements.
  - flow of bits

- Establish clocking methodology.
  - 2-cycle design: fetch, execute

- Analyze how to implement each instruction.
  - determine settings of control signals

Instruction Set Architecture

Instruction set architecture (ISA).

- Determine set of primitive instructions.
  - too narrow ⇒ cumbersome to program
  - too broad ⇒ cumbersome to build hardware

- X-TOY machine: 16 instructions.

<table>
<thead>
<tr>
<th>Instructions</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: halt</td>
<td>8: load</td>
</tr>
<tr>
<td>1: add</td>
<td>9: store</td>
</tr>
<tr>
<td>2: subtract</td>
<td>A: load indirect</td>
</tr>
<tr>
<td>3: and</td>
<td>B: store indirect</td>
</tr>
<tr>
<td>4: xor</td>
<td>C: branch zero</td>
</tr>
<tr>
<td>5: shift left</td>
<td>D: branch positive</td>
</tr>
<tr>
<td>6: shift right</td>
<td>E: jump register</td>
</tr>
<tr>
<td>7: load address</td>
<td>F: jump and link</td>
</tr>
</tbody>
</table>
Designing a Processor

How to build a microprocessor?

- Develop instruction set architecture (ISA).
  - 16-bit words, 16 TOY machine instructions
- Determine major components.
  - memory, registers, ALU, program counter
- Determine datapath requirements.
  - flow of bits
- Establish clocking methodology.
  - 2-cycle design: fetch, execute
- Analyze how to implement each instruction.
  - determine settings of control signals

Main Memory

TOY main memory: 256 x 16-bit register file.

Registers

TOY registers: extend 16 x 16-bit register file.

- Want to be able to read two registers, and write to a third in the same instructions: $R1 \leftarrow R2 + R3$.
- 3 address inputs, 2 data outputs.
- Add extra bank of muxes to bit-slice memory.
Datapath and Control

Datapath.
- Wires that transfer contents of memory, registers, and PC.
- Must accommodate for all types of instructions.

Control.
- Wires used to choreograph the flow of information on the datapath.
- Depending on instruction, different control wires are turned on.

The X-TOY Datapath

The X-TOY Datapath: Add

Before fetch:
\[ pc = 20, \text{mem}[20] = 1234 \]

After fetch:
\[ pc = 21, \text{IR} = 1234 \]

Before execute:
\[ pc = 20, \text{IR} = 1234 \]

After execute:
\[ pc = 21, \text{IR} = 1234 \]
\[ R[2] = 008C \]
The X-TOY Datapath: Jump and Link

Before fetch:
- pc = 20
- mem[20] = FF30

After fetch:
- pc = 21
- IR = FF30

Before execute:
- pc = 21
- IR = FF30

After execute:
- pc = 30
- R[F] = 21
Designing a Processor

How to build a microprocessor?

- Develop instruction set architecture (ISA).
  - 16-bit words, 16 TOY machine instructions

- Determine major components.
  - memory, registers, ALU, program counter

- Determine datapath requirements.
  - flow of bits

- Establish clocking methodology.
  - 2-cycle design: fetch, execute

- Analyze how to implement each instruction.
  - determine settings of control signals

Clocking Methodology

Two cycle design (fetch and execute).

- Use 1-bit counter to distinguish between 2 cycles.
- Why not just use 1 cycle?

Clocking Methodology

4 distinguishable events.

- During fetch phase.
- During execute phase.
- At very end of execute phase.
- At very end of fetch phase.

Can only write to register at very end of execute phase.
Control

Control: controls components, enables connections.
- Turn on certain control wires, depending on:
  - opcode, clock, conditional evaluation
- Determine orange control wires from pink ones.

Stand-Alone Registers

Instruction Register

Program Counter

ALU Control

<table>
<thead>
<tr>
<th>op</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>+, -</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>&amp;</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>^</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>&lt;, &gt;&gt;</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>input 2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Memory Control

Memory

- addr for load, store
- execute
- pc
- 16
- W Data
- R Data
- execute
- 16
- W Data

Register Control

Registers

- ALU result, load address
- pc for jal
- memory
- W
- Data Write
- Address A
- Address B
- Data A
- Data B
- Address Write
- Address A
- Address B
- 2
- W
- t
- d
- s

Pipelining

- At any instant, processor is either fetching instructions or executing them (half of circuitry is idle).
- Why not fetch next instruction while current instruction is executing?

Issues

- Jump and branch instructions change PC.
- Fetch and execute cycles may need to access same memory.

Result

- Better utilization of hardware.
- Can double speed of processor.

Layers of Abstraction

Layers of abstraction.

- Raw materials.
  - abstract switch (transistor)
  - connector (wire)
  - clock
- Logic gates.
  - AND, OR, NOT
- Combinational circuits.
  - decoder, multiplexer, majority, odd parity, adder, ALU
- Sequential circuits.
  - flip-flops, register, memory, counter
- X-TOY computer.
History + Future

Computer constructed by layering abstractions.
- Better implementation at low levels improves EVERYTHING.
- Ongoing search for better abstract switch!

History.
- 1820s: mechanical switches (Babbage's difference engine).
- 1940s: relays, vacuum tubes.
- 1950s: transistor, core memory.
- 1960s: integrated circuit.
- 1970s: microprocessor.
- 1980s: VLSI.
- 1990s: integrated systems.
- 2000s: web computer.
- Future: DNA, quantum, optical soliton, ...

The X-TOY Computer

The Big Picture

The five classic components of a computer in the von Neumann model.