Lecture A5: Building a TOY

The TOY Machine

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

What we've done.
- Written programs for the TOY machine.
- Software implementation of fetch-execute cycle.
  TOY simulator.

Our goal today.
- Hardware implementation of fetch-execute cycle.
  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.
  - ALU, memory, registers, 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).
- 16-bit words, 256 words of memory, 16 registers.
- Determine set of primitive instructions.
  - too narrow => cumbersome to program
  - too broad => cumbersome to build hardware
- TOY machine: 16 instructions.

Instructions

<table>
<thead>
<tr>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: halt</td>
</tr>
<tr>
<td>1: add</td>
</tr>
<tr>
<td>2: subtract</td>
</tr>
<tr>
<td>3: and</td>
</tr>
<tr>
<td>4: xor</td>
</tr>
<tr>
<td>5: shift left</td>
</tr>
<tr>
<td>6: shift right</td>
</tr>
<tr>
<td>7: load address</td>
</tr>
<tr>
<td>8: load</td>
</tr>
<tr>
<td>9: store</td>
</tr>
<tr>
<td>A: load indirect</td>
</tr>
<tr>
<td>B: store indirect</td>
</tr>
<tr>
<td>C: branch zero</td>
</tr>
<tr>
<td>D: branch positive</td>
</tr>
<tr>
<td>E: jump register</td>
</tr>
<tr>
<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.
  - ALU, memory, registers, 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

Arithmetic Logic Unit

TOY ALU.
- Big combinational circuit. (lecture A3)
- 16-bit bus.
- Add, subtract, and, xor, shift left, shift right, copy input 2.

Main Memory

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

Registers

TOY registers: fancy 16 x 16-bit register file.
- Want to be able to read two registers, and write to a third in the same instructions: R1 ← R2 + R3.
- 3 address inputs, 2 data outputs.
- Add extra bank of muxes for a second read port.
Designing a Processor

How to build a microprocessor?

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

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

Datapath and Control

Datapath.
- Layout and interconnection of components.
- Must accommodate all instruction types.

Control.
- Choreographs the “flow” of information on the datapath.
- Depending on instruction, different control wires are turned on.

The TOY Datapath

The TOY Datapath: Add

Before fetch: $pc = 20$, mem[20] = 1234

The TOY Datapath: Add

Before execute:
\[ \text{pc} = 21 \]

After execute:
\[ \text{pc} = 21 \]
\[ R[2] = 008C \]

The TOY Datapath: Jump and Link

Before fetch:
\[ \text{pc} = 20 \]
\[ \text{mem}[20] = FF30 \]

After fetch:
\[ \text{pc} = 21 \]
\[ \text{IR} = FF30: R[F] \leftarrow 21; \text{pc} \leftarrow 30 \]
Designing a Processor

How to build a microprocessor?

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

- Determine major components.
  - ALU, memory, registers, 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?

Can only write to register at very end of execute phase.
Designing a Processor

How to build a microprocessor?

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

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

Control

Control: controls components, enables connections.
- Input: opcode, clock, conditional evaluation. (green)
- Output: control wires. (orange)

Implementation of Control

Stand-Alone Registers
**Arithmetic Logic Unit: Implementation**

**ALU Control**

**Layers of Abstraction**

- **Layers of abstraction.**
  - Raw materials.
    - abstract switch (transistor)
    - connector (wire)
    - clock (crystal oscillator)
  - Logic gates.
    - AND, OR, NOT
  - Combinational circuits.
    - decoder, multiplexer, majority, odd parity, adder, ALU
  - Sequential circuits.
    - flip-flops, register, memory, counter
  - 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, ...
Lecture A5: Extra Notes

Memory Control

- Load, store
- Indirect load
- Subtract
- Shift left
- Load address
- Add
- AND
- XOR
- Shift right
- Branch zero
- Branch positional
- Jump and link

Register Control

- ALU result
- Load address
- PC for jal

Registers

- Data Write
- Data A
- Address Write
- Data B
- Address A
- Address B
- W