Designing a CPU

Let's build a computer!

CPU: "central processing unit"
computer: CPU + display + optical disk + metal case + power supply + ...

Last lecture: circuit that implements an adder
This lecture: circuit that implements a CPU

TOY Lite

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

Primary Components of Toy-Lite CPU

Arithmetic and Logic Unit (ALU)

Memory

Toy-Lite Registers

Processor Registers: Program Counter and Instruction Register

"Control"

Goal: CPU circuit for TOY-Lite (same design extends to TOY, your computer)
Review of Combinational Circuits

Controlled switch.

Gates.

Sum-of-products implementation of Boolean functions

Adder

Useful Combinational Circuits

Adder

Incrementer (easy, add 0001)

Bitwise AND, XOR (easy)

Decoder

Shifter (clever, but we'll skip details)

Multiplexer

Decoder. [n-bit]
- n address inputs, 2^n data outputs.
- Addressed output bit is 1; others are 0.
- Implements n Boolean functions

Decoder application: ALU!

TOY arithmetic
1: add
2: subtract
3: and
4: xor
5: shift left
6: shift right

Details:
- All circuits compute their result.
- Decoder lines AND all results.
- "one-hot" OR collects answer.
Primary Components of Toy-Lite CPU

- **Arithmetic and Logic Unit (ALU)**
- **Memory**
- **Toy-Lite Registers**
- **Processor Registers: Program Counter and Instruction Register**

"Control"

Nuts and Bolts: Buses and Multiplexers

- **Bus.** Parallel wires connecting major component.
  - Ex. Carry register bits to ALU.
  - Ex. Carry register bits to memory

- **Multiplexer (MUX).** Combinational circuit that selects among input buses.
  - Exactly one select line i is activated.
  - Copies bits from input bus i to output bus.

**CPU is a circuit:** everything is "connected" but wires that can be "on" can be selected by other wires.

A New Ingredient: Circuits With Memory

- **Combinational circuits.**
  - Output determined solely by inputs.
  - Ex: majority, adder, decoder, MUX, ALU.

- **Sequential circuits.**
  - Output determined by inputs and current "state".
  - Ex: memory, program counter, CPU.

**Ex.** Simplest feedback loop.
- Two controlled switches A and B, both connected to power, each blocked by the other.
- State determined by whichever switches first.
- Stable.

**Aside.** Feedback with an odd number of switches is a **buzzer** (not stable).
Doorbell: buzzer made with relays.
SR Flip-Flop

• Two cross-coupled NOR gates
• A way to control the feedback loop.
• Abstraction that "remembers" one bit.
• Basic building block for memory and registers.

Caveat. Timing, switching delay.

Memory Overview

Computers and TOY have several memory components.
• Program counter and other processor registers.
• TOY registers (4 10-bit words in Toy-Lite).
• Main memory (16 10-bit words in Toy-Lite).

Implementation.
• Use one flip-flop for each bit of memory.
• Use buses and multiplexers to group bits into words.

Access mechanism: when are contents available?
• Processor registers: enable write.
• Main memory: select and enable write.
• TOY register: dual select and enable write

need to be able to read two registers at once

Processor register Bit

Processor register bit. Extend a flip-flop to allow easy access to values.

Memory Bit Interface

Memory and TOY register bits: Add selection mechanism.
Memory Bit: Switch Level Implementation

Memory and TOY register bits: Add selection mechanism.

Processor Register

Processor register: don't confuse with TOY register
- Stores k bits.
- Register contents always available on output bus.
- If enable write is asserted, k input bits get copied into register.

Ex 1. TOY program counter (PC) holds 8-bit address.
Ex 2. TOY instruction register (IR) holds 16-bit current instruction.
**Memory Bank**

1. Memory bank:
   - Bank of n registers; each stores k bits.
   - Read and write information to one of n registers.
   - Address inputs specify which one.
   - Addressed bits always appear on output.
   - If write enabled, k input bits are copied into addressed register.

2. Ex 0 (for lecture). 4-by-6
   - (four 6-bit words)

3. Ex 1. Main memory bank.
   - TOY: 256-by-16
   - TOY-Lite: 16-by-10

4. Ex 2. Registers.
   - TOY: 16-by-16
   - TOY Lite: 4-by-10
   - Two output buses.

---

**Memory: Interface**

**Memory: Component Level Implementation**

Decoder plus memory selection: connect only to addressed word.

**Memory: Switch Level Implementation**
**TOY-Lite Memory**

- 16 10-bit words
  - input connected to registers for "store"
  - output connected to registers for "load"
  - addr connect to processor Instruction Register (IR)

**Toy-Lite Registers**

- 4 10-bit words
  - Dual-ported to support connecting two different registers to ALU
  - Input MUX to support input connection to ALU, memory, IR, PC

---

**Primary Components of Toy-Lite CPU**

- ✓ ALU
- ✓ Memory
- ✓ Registers
  
  ✓ Processor Registers: Program Counter and Instruction Register

  "Control"

---

**How To Design a Digital Device**

How to design a digital device.

- Design interface: input buses, output buses, control wires.
- Determine components.
- Determine datapath requirements: "flow" of bits.
- Establish control sequence.

**Warmup.** Design a program counter (3 devices, 3 control wires).

**Goal.** Design TOY-Lite computer (10 devices, 27 control wires).
Program Counter: Interface

**Counter.** Holds value that represents a binary number.
- Load: set value from input bus.
- Increment: add one to value.
- Enable Write: make value available on output bus.

**Ex.** TOY-Lite program counter (4-bit).

Program Counter: Components

**Components.**
- Register.
- Incrementer.
- Multiplexer (to provide connections for both load and increment).

Program Counter: Datapath and Control

**Datapath.**
- Layout and interconnection of components.
- Connect input and output buses.

**Control.** Choreographs the “flow” of information on the datapath.
Program Counter: Datapath and Control

Datapath.
• Layout and interconnection of components.
• Connect input and output buses.

Control. Choreographs the “flow” of information on the datapath.

Primary Components of Toy-Lite CPU

✓ ALU
✓ Memory
✓ Toy-Lite Registers
✓ Processor Registers: Program Counter and Instruction Register

How To Design a Digital Device

How to design a digital device.
• Design interface: input buses, output buses, control wires.
• Determine components.
• Determine datapath requirements: “flow” of bits.
• Establish control sequence.

Warmup. Design a program counter (3 devices, 3 control wires).

Next. Design TOY-Lite computer (10 devices, 27 control wires).
CPU is a circuit.

Interface: switches and lights.
- set memory contents
- set PC value
- press RUN
- [details of connection to circuit omitted]
Instructions determine datapaths and control sequences for execute

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Datapath</th>
<th>Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>halt</td>
<td>Memory[PC] to IR</td>
<td>increment PC</td>
</tr>
<tr>
<td>add</td>
<td>Memory[PC] to IR</td>
<td>increment PC</td>
</tr>
<tr>
<td>subtract</td>
<td>IR opcode to control</td>
<td>two registers to ALU</td>
</tr>
<tr>
<td>and</td>
<td>IR to addr MUX</td>
<td>memory to register MUX</td>
</tr>
<tr>
<td>xor</td>
<td>IR opcode to control</td>
<td>ALU to register MUX</td>
</tr>
<tr>
<td>shift left</td>
<td>Memory[PC] to IR</td>
<td>increment PC</td>
</tr>
<tr>
<td>shift right</td>
<td>Memory[PC] to IR</td>
<td>increment PC</td>
</tr>
<tr>
<td>load address</td>
<td>IR opcode to control</td>
<td>two registers to ALU</td>
</tr>
<tr>
<td>load</td>
<td>IR to addr MUX</td>
<td>memory to register MUX</td>
</tr>
<tr>
<td>store</td>
<td>Memory[PC] to IR</td>
<td>increment PC</td>
</tr>
<tr>
<td>load indirect</td>
<td>IR opcode to control</td>
<td>ALU to register MUX</td>
</tr>
<tr>
<td>store indirect</td>
<td>IR to addr MUX</td>
<td>memory to register MUX</td>
</tr>
<tr>
<td>branch zero</td>
<td>IR opcode to control</td>
<td>two registers to ALU</td>
</tr>
<tr>
<td>branch positive</td>
<td>IR to addr MUX</td>
<td>memory to register MUX</td>
</tr>
<tr>
<td>jump register</td>
<td>IR opcode to control</td>
<td>ALU to register MUX</td>
</tr>
<tr>
<td>jump and link</td>
<td>Memory[PC] to IR</td>
<td>increment PC</td>
</tr>
</tbody>
</table>
Control. Each instruction corresponds to a sequence of control signals.

Q. How do we create the sequence?
A. Need a “physical” clock.

Solution 1: Use some other technology

Solution 2: Use a buzzer [need sufficiently long cycle to cover CPU switching]

How much does it Hertz?

Frequency is inverse of cycle time.
- Expressed in hertz.
- Frequency of 1 Hz means that there is 1 cycle per second.
  - 1 kilohertz (kHz) means 1000 cycles/sec.
  - 1 megahertz (MHz) means 1 million cycles/sec.
  - 1 gigahertz (GHz) means 1 billion cycles/sec.
  - 1 terahertz (THz) means 1 trillion cycles/sec.

Clock.
- Fundamental abstraction: regular on-off pulse.
  - on: fetch phase
  - off: execute phase
- “external” device.
- Synchronizes operations of different circuit elements.
- Requirement: clock cycle longer than max switching time.

Clocking Methodology

Two-cycle design.
- Each control signal is in one of four epochs.
  - fetch [set memory address from pc]
  - fetch and clock [write instruction to IR]
  - execute [set ALU inputs from registers]
  - execute and clock [write result of ALU to registers]
Control. Circuit that determines control line sequencing.

CPU is a circuit, driven by a clock.

Switches initialize memory, PC contents.

Clock ticks
- **fetch** instruction from memory[PC] to IR
- increment PC
- **execute** instruction

[details of instruction execution differ]

- **fetch** next instruction
- ...

That's all there is to it!
Real Microprocessor (MIPS R10000)

Layers of Abstraction

<table>
<thead>
<tr>
<th>Abstraction</th>
<th>Built From</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract Switch</td>
<td>raw materials</td>
<td>transistor, relay</td>
</tr>
<tr>
<td>Connector</td>
<td>raw materials</td>
<td>wire</td>
</tr>
<tr>
<td>Clock</td>
<td>raw materials</td>
<td>crystal oscillator</td>
</tr>
<tr>
<td>Logic Gates</td>
<td>abstract switches,</td>
<td>AND, OR, NOT</td>
</tr>
<tr>
<td></td>
<td>connectors</td>
<td></td>
</tr>
<tr>
<td>Combinational Circuit</td>
<td>logic gates, connectors</td>
<td>decoder, multiplexer, adder, ALU</td>
</tr>
<tr>
<td>Sequential Circuit</td>
<td>logic gates, clock, connector</td>
<td>flip-flop</td>
</tr>
<tr>
<td>Components</td>
<td>decoder, multiplexer, adder, flip-flop</td>
<td>registers, ALU, counter, control</td>
</tr>
<tr>
<td>Computer</td>
<td>components</td>
<td>TOY</td>
</tr>
</tbody>
</table>

History + Future

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

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

Overview

What is COS 126?
- Broad, but technical, intro to CS.
  No prerequisites, intended for novices.

Goals.
- Demystify computer systems.
- Empower you to exploit available technology.
- Build awareness of substantial intellectual underpinnings.

Topics.
- Programming in Java.
- Machine architecture.
- Theory of computation.
- Applications to science, engineering, and commercial computing.