Microarchitecture

- Hardware components in a digital circuit or system
  - What they are
  - How they are connected
- Also called datapath

Main components

- (Combinational) logic
  - Functions (adders, multipliers, shifters)
  - MUXes
- Wiring
  - Point-to-point
  - Busses
- Storage
  - Small scale storage (registers)
  - Large scale storage (memory)
- Control (contains storage, logic and wiring)

Abstraction

- Logic and wiring are instantaneous
  - Output is always a function of the values on input wires
  - If input changes, the change is “processed immediately”
- Storage elements are timed
  - Clock – a special signal that determines this timing
  - Storage can be updated only at the tick of the clock
- What happens between ticks?
  - The “current” values are processed by logic and wiring to produce values …
  - … that will be used to update at the “next tick”
Combinational Logic

- A digital circuit that computes a function of the inputs.

Examples:
- Adder: takes $X$ and $Y$ and produces $X + Y$
- AND: takes $X$ and $Y$, produces bitwise and
- NOT: takes $X$ and $Y$ and produces $\neg X$
- MUX: takes three inputs, $X$, $Y$ and $s$ (the last one is 1-bit) and produces (note that this is C-syntax, not the RTN that we will show later) $(s==0) \ ? \ X \ : \ Y$

Wires and Busses

- Wires are (almost) just like electrical wires
  - Directional (arrows)
  - May have a “thickness” number of bits of data: e.g., the adder output is 16-bits in LC-3

Busses:
- Shared wires
- Anyone can read at all times
- Write is via arbitration (control signals to decide who gets to write on the bus)

Storage Elements

- Large scale storage (memory): view it like an array
  - Address + Data
- Small scale storage (registers):
  - Programmer-visible registers: $R0 \ldots R7$
- Special purpose registers:
  - PC, IR, PSR (processor status register), MAR, MDR

Memory

- Processor issues commands to memory, who responds
  - Mem.EN (memory enable): he’s talking to you
  - Mem.RW: here’s what I want you to do
- Two special registers
  - Memory Address Register (MAR): only processor writes to this
  - Memory Data Register (MDR): both processor/memory can write to this
  - the processor arbitrates

If Mem.EN and if Mem.RW==0, (i.e., read) the memory copies the value at address MAR into the MDR, otherwise copy the contents of MDR into Mem[MAR]

Registers

Every register
- is connected to some inputs
- has a special “load” signal
  - If load signal is 1 at the next clock tick the input is stored into the register
  - Otherwise, no change in register contents

(LD.PC) ? PC ← PC+1

Register Transfer Notation

- Compact, “program-like” notation
- Describe what happens in the datapath
- One or more transfers per clock tick
  - one line = one clock tick
- Two columns:
  - Write the desired transfers
  - List control signals to “effect the transfer”
- Let’s move on to LC3-Viz (special thanks, Joe Arnett)
- Corrections
  - BR uses IR[8:0] instead of IR[10:0] for the PC offset

RTN/LC3-Viz Conventions

- Signals indicated must be asserted before the clock tick in order for the indicated transfer to occur. Sequence is:
  - Signals are asserted
  - Clock tick arrives, and causes the transfer
- In an RTN transfer, on either the right hand side (rhs), or left hand side (lhs)
  - Mem[x] is the memory at address x
  - Mem(MAR) is the memory at address that is in the MAR
  - Reg[x] is Register number x
RTN Conventions

- An RTN transfer is of the form:
  \[ \text{LHS-location} \leftarrow \text{RHS-expression} \]
- The LHS location may be a memory or a specific register or the \(x\)-th register
- The RHS expression is:
  - named registers, e.g., \(\text{Reg[3]}\)
  - memory locations e.g., \(\text{Mem[MAR]}\)
  - simple expressions \(\text{PC+1, Reg[src] + Reg[dst]}\)

How does the LC-3 fetch an instruction?

1. Transfer the PC into MAR
   - Cycle 1: MAR \(\leftarrow\) PC
   - # Transfer the PC into MAR
   - # LD.MAR, GatePC
2. Read memory; increment PC
   - Cycle 2: MDR \(\leftarrow\) Mem[MAR]; PC \(\leftarrow\) PC+1
   - # LD.MDR, MDR_SEL, MEM.EN, LD.PC, PROM
3. Transfer MDR into IR
   - Cycle 3: IR \(\leftarrow\) MDR
   - # Transfer MDR into IR
   - # LD.IR, GateMDR

How does the LC-3 decode the instruction?

- Special decode step (controller makes decision, no clock cycle is wasted since it only involves logic)
- # No visible signal is active

How does the LC-3 execute a NOT instruction?

- Src register contents are negated by ALU and result is stored in dst register
- Cycle 4: Reg[dst] \(\leftarrow\) -Reg[src]; CC \(\leftarrow\) Sign(-Reg[src])
  - # LD.REG, DR = dst, GateALU, ALU = ~, SR1 = src, LD.CC
Other instructions

- Every instruction is a sequence of transfers
- Every one has the same first three cycles (instruction fetch)
- Every one takes (at least one) additional cycle
- Some take even more more
- Each one effected by a specific set of control signals
- The Controller is responsible for generating the correct signals in the appropriate cycle
- Reminder
  - Logic is instantaneous,
  - Storage (transfers) are on clock ticks