Computer Organization and Architecture

Conrad A. Mearns

February 9, 2017

Turing Machines

1. Turing's Thesis: Every computation can be represented with a Turing Machine.
2. Turing Machine: A mathematical model of a device that can preform any computation.
3. Universal Turing Machine: A machine to implement any and all Turing Machines.

Beyond models, real world constraints include time, financial cost, power, security, thermal dissapation, space, etc.

Bits, Data Types, and Operators

The electro-magnetic field is not digital, yet all of modern computing is represented digitally. To compromise, 0 is a representation of the absence of voltage and 1 is a representation of the presence of voltage.

0V 0.5V "Illegal" 2.4V 2.9V

Signed Binary Arithmetic

Binary, without the addition of extra mathematical symbols, can only represent positive whole integers. Signed numbers like -5 require a formal system of representation in order to be used. A binary number can represent 2^n values for n bits. The objective of signed numbers is to partition half of those values for negative number representation $(-2^{n-1} - 1 \to -1)$ and the other half to positive numbers $(1 \to 2^{n-1})$ while leaving zero and potentially one other number available.

1. Sign-Magnitude
   The most significant bit is 0 for positive values and 1 for negative values.
   $00101 = 5$ and $10101 = -5$
2. One's Complement
   All bits are inversed to represent negative numbers. Like Sign-Magnitude, the most significant bit will tell you whether a number is positive or negative.
   $00101 = 5$ and $11010 = -5$

3. Two's Complement (currently in use)
   For each positive number A, it's negative number (B) satisfies the equation $A + B = 0$ when the final carried bit is dropped. To get this number, take the One's Complement of A and add 1. $00101 = 5$ and the One's Complement is 11010. So the Two's Complement is 11011. As proof, $00101 + 11011 = 100000$ but the last carried one is dropped, leaving 00000.

Arithmetic and Logical Operations

Arithmetic operations

1. Addition
   Just addition, regardless if signed or not. Ignore the final carry-out.

2. Subtraction
   First negate the second operand ($5 \to -5 for example$), then use addition.

3. Sign Extension
   To add numbers, they must have the same number of bits. This is because of signed numbers, and storage.

Overflow occurs when

1. Signs of the operands are the same

2. The sign of the sum is different

The issue can be tested for by examining the most significant bit's sign between the operands and the result.
Logical operations

1. AND
   The result is true if and only if both operands are true.
   Useful for clearing bits, a mask of 1's signify keep.

2. OR
   The result is true if either operand is true.
   Useful for setting bits. 1's in the second operand copy to the result.

3. NOT
   The result is true if and only if the operand is false.

Each operation is executed on each bit individually.

Fractions, Floating, and Fixed-Point Values

A "binary" point is abstractly added to the value. To the left of the point, each bit is worth is 2^{-n} where n is the place left of the point. For example, 101.11. For large numbers, we use scientific notation. $Sign * (Fraction * 2^{Exponent})$
IEEE 754 Floating Point Standard for 32-bits signifies 1 bit for sign, 8 bits for

the exponenant, and 23 bits for the fraction.
$1 - 01111110 - 10000000000000000000000 = -1.5 * 2^{???}$

Other Date Types

1. Single characters use ASCII to map 128 characters to 7-bit code.
2. Text strings are sequences of characters often with a NULL to terminate. No hardware support.
3. Images are arrays of images. Often has hardware support.
4. Sound is a sequence of fixed-point numbers.
5. Other data types my be defined abstractly by us and interpreted as needed.

Transistors

Each transistor is a digital switch. When combined in different circuits, transitor combinations can emulate certain logical operations such as AND, OR, and NOT. These can then be combined to create adders, multiplexers, decoders, and other farther complex structures.
Gordon Moore, an early founder of Intel, hypothesized that transistor count would double every two years. This is now known as Moore's Law.
Transitors have 3 wells. A well is a homogeneous material that is either more or less positive than negative. A negative well is denoted n and a positive well is denoted p. The gate in a transitor creates a field to allow or dissallow charge to migrate across wells.
n-Type Transistor (npn)
Two n wells inside a large p substrate.

- When gate is tied to GND, the switch is open.
  No current flows from the source to the drain.
  '0 state'
- When gate is tied to voltage the switch is closed.
  Current flows from the source to the drain.
  '1 state'

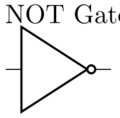
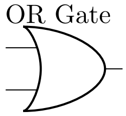
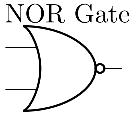
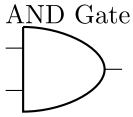
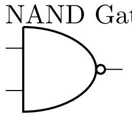
p-Type Transistor (pnp)
Two p wells inside a large n substrate.

- When gate is tied to GND, the switch is closed.
  Current flows from the source to the drain.
  '1 state'
- When gate is tied to voltage the switch is open.
  No current flows from the source to the drain.
  '0 state'

Complementary Metal Oxide Semiconductor Circuits(CMOS)

- NOT Gate (Inverter)
- NOR Gate (NOT OR, Serial on top, Parallel on bottom)
- OR Gate (NOR + NOT)
- NAND Gate (NOT AND, Parallel on top, Series on bottom)
- AND Gate (NAND + NOT)

Simplified Gates

- NOT Gate
  
- OR Gate
  
- NOR Gate
  
- AND Gate
  
- NAND Gate
  

Other Circuits

- 2-Bit Decoder
  Uses 4 AND Gates with different inverter configurations on each. It maps 2 inputs to 4 outputs.
- Multiplexer
  Uses 4 inputs and interprets to 1 output. A selector input of two wires will change the output from 4 AND Gates.

- Full Adder
  Capable of adding two bits with carry-in. Produces a one-bit sum with carry-out

Logical Completeness

Can complete any truth table with just AND, OR, NOT. First, mark every output row that has a truth value of true. Draw an OR gate at the bottom to accept all true outputs. Connect AND Gates to the OR Gate and have every input connect to each AND gate. For each AND Gate, configure the inputs with invertes so that each AND Gate emulates a truth table row.

Combinational vs Sequential Circuits

Combinational Circuit
always produces the same output for a given set of inputs
Sequential Circuit
Stores information
Output depends on stored information (state) plus input

R-S Latch

R is used to 'Reset' or clear the element - set it to zero.
S is used to 'set' the element - set it to one.
If both R and S are one, the output could be either one or zero. To assert one of the inputs, use Active Low Logic.

Gated D Latch

Based upon R-S Latch and has 2 inputs. D (for data) and WE (write enabled). Two AND gates feed into the R and S inputs of an R-S latch, and input is only sent to the R-S latch when WE is asserted.
WE = 1 $\rightarrow$ latch is set to the value of D. S = NOT(D), R = D
WE = 0 $\rightarrow$ latch holds previous value. S = R = 1

Register

Side by side Gated D latches that share a single WE line, but has seperate data lines.
A register holds a n-bit value, controlled by a common WE.

Memory

Address is taken as multple inputs, that input passes to an address decoder which uses AND gates to specify what memory location can be written to, when WE is asserted.
Not the most expensive, and more transistors are needed in greater desnity.
Address decoder
Word Select Line
Word Write Enable

Representing Multi-bit Numbers

Number bits from right to left for convention. Use brackets to denote range.
D[l:r] denotes bit l to bit r from left to right in register D
May also see "A¡14:9¿"

State Machine

Finite State Machine with Datapath (FSMD)
Controller / Data Path
Another type of combinational logic with storage. It ”Remembers” states and changes it’s output(s) are based on inputs and the state machine’s current state. This type of circuit is the heart of the controller in a CPU.
Combinational vs Sequential: Combinational types depend only on the values. Sequential types sctrictly depend on the order of values inputted.
The state of a system is a snapshot of all the relevant elements of the system at the moment the snapshot is taken.
State diagrams are directed graphs that show how actions change states.

1. A finite number of states
2. A finite number of external inputs
3. A finite number of external outputs
4. An explicit specification of all state transitions
5. An explicit specification of what determines each external value

Clock cycles are used in digital circuits to trigger state transitions. A single cycle is when the value changes between ’1’ and ’0’ fully. Transitions can be triggered with edge triggered logic, or level triggered logic.
Storage for state machines can be accomplished using a Master-Slave flipflop.

From Logic to Datapath

The datapath of a computer is all the logic used to process information.

- Combinational Logic
  Decoders – convert instructions into control signals
  Multiplexers – select inputs and outputs
  ALU (Arithmetic and Logic Unit) – operations on data
- Sequential Logic
  State machine – coordinate control signals and data movement
  Registers and latches – storage elements

von Neumann Machine / Model

Basic structure of machine that is the most common, even today.
a memory, containing instructions and data
a processing unit, for preforming arithmetic and logical operations
a control unit for interpreting instructions

Harvard Model

Refinement of von Neumann Model.
seperate memory for programs and data
Both models are sequential and synchronous. Programs are both interpreted by a control unit.

- IR - Instruction Register

- PC - Program Counter
- ALU - Arithmetic and Logical Unit
- MAR - Memory Address Register
- MDR - Memory Data Register
- PMEM - Program Memory (Harvard Model, effectively Read-Only)
- DMEM - Data Memory (Harvard Model)

Memory

$2^k * m$ array of stored bits.
Address - unique k-bit identifier of location
Contents - m-bit value stored in location
Basic Operations:
LOAD - Read a value from a memory location
STORE - Write a value to a memory location

Interface to Memory

To LOAD a location (A)

- Write the address (A) into the MAR
- Send a 'read' signal to the memory
- Read the data stored in MDR

To WRITE a value (X) to a location (A)

- Write data (X) to the MDR
- Write the address (A) to the MAR
- send a 'write' signal to the memory

Processing Unit

Functional Units - add, multiply, square root
Registers - Small temp storage, operands and results of functional units
Data Word Size - number of bits normally processed by ALU in one instruction. Also the width of registers