Announcements

- Midterm #2 is Thursday November 9

What does the Midterm Cover?

- Mano Chapters 1-5
  - anything from lectures
  - material from the homework
- What's not on this Midterm?
  - anything after Chapter 5
  - details about how to build ALU and control lines for Mano Computer (but ALU's from Chapter 4 will be covered)

How do we represent real numbers?

- Each number has two parts
  - mantissa (represents a number between -1 and 1)
  - exponent (designates the position of the decimal point)
  - number = m r^e where r is the radix
    - 6132.789 = +0.6132789 x 10^4
- A number is normalized if the most significant digit of mantissa is non-zero.
- Computers normally use a radix of 2
  - 1001.11 = .1001110 x 2^4
  - 10.5 = 1010.1 or 10101 x 2^4
  - 17.451 = 111.011100 = 1110111 x 2^3

Other Ways to Represent Information

- Other Numeric Codes
  - Gray Codes
  - Excess-3 codes
  - 2 of 5 code
  - parity (use one extra bit to check for an error)
    - odd parity - any info has an odd # of one's
    - even parity - any info had an even # of one's
- ASCII
  - For characters

ALU

- Math Functions
  - addition
  - subtraction (a-b --> a + b' + 1)
  - use full addres for each bit position
- Shift Operations
  - logical, circular, arithmetic
- Logic operations
  - AND, OR, NOT
  - used for selective bit operations
    - set, complement, clear
- Often Use Mux to Implement an ALU

Stored Program Computer

- Can build “stored program” FSM
  - use EPROM to hold transitions
  - use register to store current state
- Stored Program Computer
  - instructions and data are stored in memory
  - instructions
    - standardized set of operations on registers and memory
    - a sequence of micro-instructions defines the instructions
    - normally execute instructions in memory order
      - unless we have a branch instruction

Instruction Format

- Instruction consists of several fields
  - op code (what to do)
  - operand (where to do it)
    - can be registers or memory
  - addressing mode
    - indirect memory references
- Fixed size of instruction limits
  - # of instructions (different bit patterns for op-codes)
  - range of memory locations (width of address field)

- Use a collection of registers to hold:
  - AC Accumulator
  - DR Data Register (holds memory operands)
  - AR Address register (12 bits)
  - IR Instruction Register
  - PC Program Counter -address of instruction (12 bits)
  - TR Temporary register
  - INPR Input Register (8 bits)
  - OUTR Output Register (8 bits)
- In addition to registers and memory:
  - Have a bus to move information around
  - ALU to compute results
  - Controller to make it all work

Examples of Micro-Instructions

- AR <- IR
  - S = 101, LD_{AR}=1
- PC <- PC + 1
  - INR_{PC}=1
- DR <- AC, AC <- DR
  - S=100, LD_{DR} =1, ALU (F=A), LD_{AC}=1
- Each clock pulse is one line of RTL
- Several things can happen on one clock pulse

Executing Instructions

- List of instructions is called a program
  - instructions are stored in memory
- Computer executes next instruction in the program
  - branch instruction changes the next instruction to execute.
  - programs start execution at memory location 0.
  - one instruction at a time is executed.
- State
  - memory and registers are assumed to start empty (all 0's)
  - after executing an instruction any change to memory or registers remains.
- To keep track of steps in executing an instruction
  - use a micro-sequencer
  - really just a fancy name for a counter with clear

- Fetch and Decode steps

  T0: AR <- PC

  T1: IR <- M[AR], PC <- PC + 1

  T2: D0,….D7 <- decode(IR(12-14)),

  AR <- IR (0-11), I <- IR(15)

- Next Execution machine specific part of instruction
  - for memory reference instructions
    - D7’ I T3: AR <- M[AR]
- Subroutine calliing
  - BSA Instruction can be used to call a subroutine
  - BUN Indirect is used to return
- Interrupts
  - like a hardware subroutine

Interrupt Cycle

- Setting the Flag
  - $T_0' T_1' T_2'$ (IEN) (FGI + FGO): R <- 1
- Interrupt Cycle
  - R T_0: AR <- 0, TR <- PC
  - R T_1: M[AR] <- TR, PC <- 0
  - R T_2: PC <- PC + 1, IEN <- 0, R <- 0, SC <- 0
- Modified Fetch and Execute
  - R' T_0: AR <- PC
  - R' T_1: IR <- M[AR], PC <- PC + 1
  - R' T_2: $D_0,\ldots D_7$ <- Decode(IR(12-14)), AR <- IR(0-11), I<-IR(15)

Change

What is Required to Build the Computer?

- A memory with 4096 words each 16 bits wide
- Nine registers:
  - AR, PC, DR, AC, IR, TR, OUTR, INPR, SC
- Seven Flip-flops: I,S,E,R,IEN,FGI,FGO
- Two decoders:
  - 3x8 and 4x16
- a 16 bit bus
- control logic gates
- added and logic unit