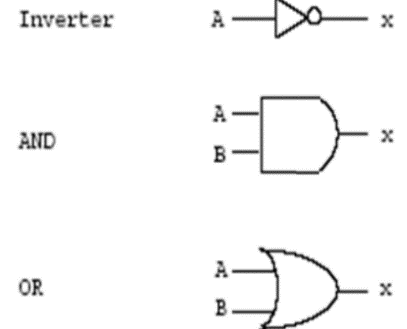
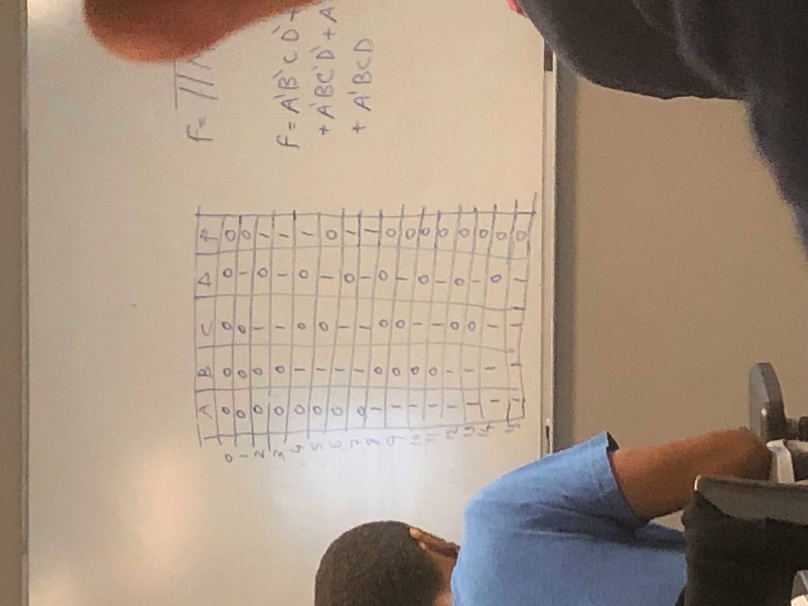
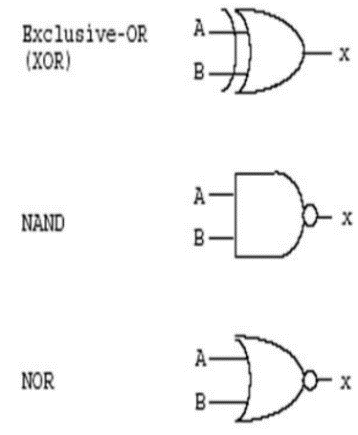
**Computer Architecture Notes**

8-26

* Main components of computer are CPU, RAM, and IOP
* CPU – made up of ALU, control unit and registers
* Components communicate with each other through electronic wires called buses

8/28

* Combinational Circuit consists of logic gates, takes binary input and creates binary output
* Sequential Circuit is a sequential circuit with memory and clock added
* Boolean function returns a Boolean output with multiple inputs, + is or, \* is and, ‘ is not
* Multiple input XOR returns 1 only when odd number of inputs are 1
* Boolean algebra theorems:
  + Idempotency: A + A = A, (A\*A) = A
  + Null Element: (a + 1) = 1, (a\*0) = 0
  + Involution: (a’)’ = a
  + DeMorgan’s: (a + b)’ = a’\*b’, (a\*b)’ = a’ + b’
  + A + A’B = A + B
* Truth table gives all possible combinations of inputs and their outputs for a Boolean function
  + Truth table rule: current column all 0s before 1s, left most variable first

Prove that:

* Ab + ab’ = a

A(b + b’)

A \* 1

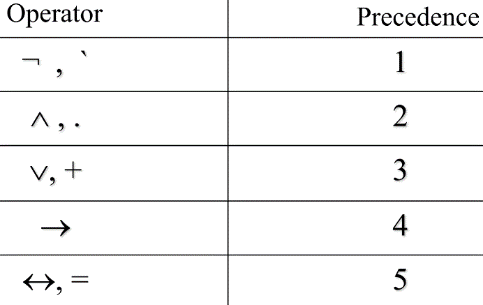
A

* (a + b)(a + b’) = a

A+ ab’ + ab + b\*b’

A + ab’ + ab

A\*1 + ab’ + ab

A(1 + b’ + b)

A\*1

A

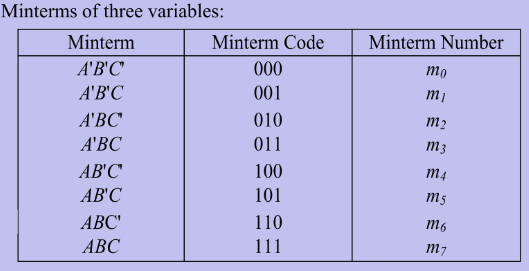
* (a + bc)’ = (a’(bc)’)

A’(b’ + c’)

A’b’ + a’c’

(a + bc)’

Given f(A, B, C, D) = sum(m(2,3,4,6,7)), construct truth table, express f in maxterms, simplify f:

F = TTM(0, 5, 8, 9, 10, 11, 12, 13, 14, 15)

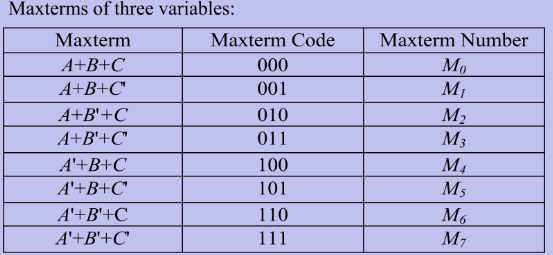
F = A’B’CD’ + A’B’CD + A’BC’D’ + A’ BCD’ + A’BCD

Truth table of all possibilities of ABCD inputs and F being 1 on row 2,3,4,6 and 7

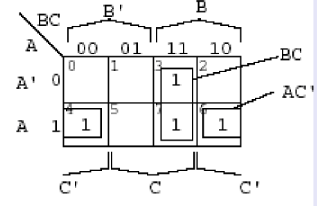
F = πM(0,1,5,8,9,10,11,12,13,14,15)

F = inputs in which F is 1 (row 2,3,4,6,7)

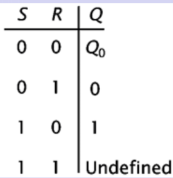
F = A’B’CD’ + A’B’CD + A’BC’D’ + A’BCD’ + A’BCD = A’B’C + A’CD + A’BD’

* Literal – a variable (complemented or not)
* Product term – literals ANDed together
  + Minterm – product term where all variables appear once
* Sum term – literals ORed together
  + Maxterm – sum term where all variables appear once
* Sum of Products (SOP) – ORing product terms
  + Canonical SOP – sum of minterms only
    - Ex. F(A,B,C) = A’BC + AB’C’ + A’B’C

**Karnaugh Maps**

* Kmap – a diagram made up of squares, with each square representing one minterm
  + 1s where minterm function returns 1, 0s for maxterm function
  + Map blocks to function takes the variables that the block squares have in common
    - Blocks of 1s can have 2n squares, less blocks the better
    - Squares can be reused for blocks with unused squares
  + Arranged so that adjacent squares only differ by one variable
  + Sides are adjacent too
  + Ex. F(A,B,C) = Σ(3,4,6,7) = Π(0,1,2,5) = BC + AC’

**Digital Components**

* Half-adder – performs the most basic digital arithmetic operation, the addition of 2 bits
  + Uses 2 bits for output (sum and carry)
* Full-adder – adds multi-bit inputs, uses a carry bit in processing
* Memory Devices – latch and flip flop edit memory by triggering pulses
  + D flip-flop – output is same as input when clock is high and latch is 1
    - Some have clear and set capabilities
    - Q(t + 1) = D
  + SR flip-flop – set and reverse inputs, only one can be active at once
    - Q(t + 1) = S + R’\*Q(t)
  + JK flip-flop –SR flip-flop version with clock, more popular
    - If JK are both 1, then instead of undefined, Q is toggled
    - Q(t + 1) = K’\*Q(t) + JQ’(t)
  + T(toggle) flip-flop – toggle Q if T input is 1, no clock
    - Q(t + 1) = T\*(Q(t))’ + T’\*Q(t)
  + 4 flip storage device which connected signals is a register
* Sequential circuit – combination circuit made of flip-flops and gates, clock synchronizes

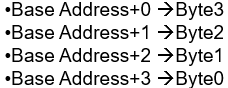
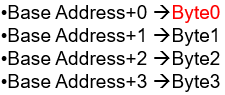
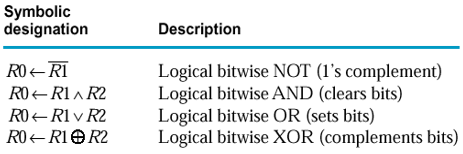
**Digital Circuits**

* Transistor – basic electronical component in digital systems, acts as on/off switch
  + Complementary Metal Oxide Semiconductor (CMOS) – data transmitted through voltage
    - 2 types: nMOS conducts if gate = 1. pMOS conducts if gate = 0
* Counters – increment or decrement by 1
  + Ripple (asynchronous) counters – flip flop output is source for triggering other flip flops
    - Delay between responses of flip flops creates ripple effect of output
  + Synchronous counters – all flip flops process at the same time
    - Most counters are synchronous in computers
  + Up counters (0000 to 1111) and down counters (1111 to 0000)
* Combinational Circuits
  + Decoder – converts binary code of n bits to a max of 2n outputs
    - Decoder is enabled when E input = 0 (otherwise all outputs are active/=1)
    - Decoders can be combined to form larger decoder with same inputs
  + Encoders – do opposite of decoders, take 2n or less input bits and makes n outputs
    - Priority Encoder – works when more than 1 input is active, output is highest input
  + Multiplexer (MUX) – multiple input bits with 1 output
    - Can combine multiplexers to make a bigger one
    - Like decoder because both decode minterms
    - Different because can have n input bits instead of 2n, has 1 output instead of n
  + Demultiplexer – perform inverse of a multiplexer
    - Receives info from a single line and transmits to one of 2n possible output lines
    - N selection inputs select the specified output
* Storage Components – store data and do simple data transformations
  + RAM contains 2 types: static and dynamic, both keep data unless power is lost
    - SRAM – very fast and expensive, used inside processors like cache
    - DRAM – main memory, longer access time but cheaper, allows huge memory capacity
      * Data is constantly rewritten/refreshed, or else is lost
  + Registers – group of flip-flops where each flip-flop stores 1 bit
    - Flip-flops hold data and combinational circuits control transfer of data to register
    - Computer clock sends continuous pulses, some pulses change register data
    - If the control signal LOAD = 1, new values is loaded, if = 0 then stays the same
    - Shift register – can shift bits in one or both directions with flip-flops in series
      * 4:1 selector register – takes 2 inputs and can load data, shift left and right

**Data Representation**

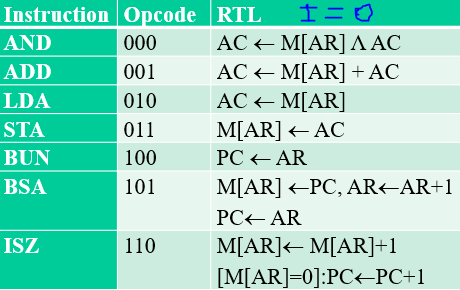
* Binary Coded Decimal (BCD) – assigns a binary nibble to each decimal symbol 0-9
* Grey Code – like binary but with changes in a different structure
  + Only 1-bit changes with increment for efficiency
* ASCII – American Standard Code for Information Interchange
  + 7-bit mapping of binary to chars, 8th for parity
  + Odd parity bit is 1 if total 1s (including parity bit) is odd, even parity if even
* Binary subtraction uses complements for efficiency
  + Complement – difference between that number and its base, ex. 10s complement of 6 is 4
    - 8 – 6 = 8 + complement (-6) = 8 + 4 = 2, carry is ignored
    - 10s complement of 546700 = 453300, 9s complement of 546700 = 453299
    - 1s complement flips all bits
      * For 1s complement addition, add numbers normally, then add the carry to the lowest value bit
      * Subtraction is same as addition, except first step is getting complement of 2nd number
    - 2s complement is used a lot in computers for binary subtraction
      * Inverts all bits then adds 1 or start from right and keep all bits the same until first 1 (invert after first 1)
      * 2s complement addition ignores final carry
      * Subtraction is same steps as 1s complement
    - Subtraction of unsigned numbers using r’s complement steps M - N:
      * Take complement of N
      * If M >= N, add M and NC, and ignore carry
      * If M < N, add M and NC, take r’s complement of sum, and add negative sign
* Signed Numbers can be represented by signed magnitude, 1s complement, or 2s complement
  + For all, left-most bit indicates sign (0 = positive, 1 = negative)
  + 1s complement with signed numbers results in the sign switched
  + Fixed-Point Representation – binary point always in one position
  + Floating-Point representation – uses a second register to store a number indicating the position of the decimal

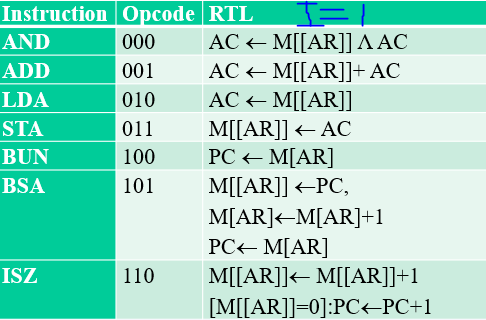
**Ch4 Microoperations**

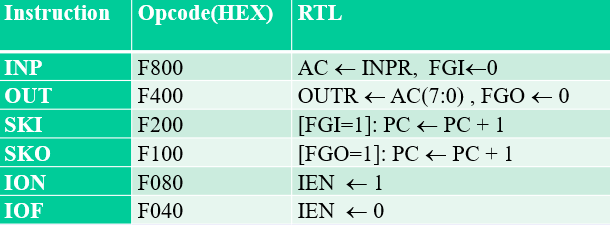
* Processor hardware: registers, ALU, bus
  + Datapath – registers and logic circuits where data flows
  + Control – stream of bits (sequence of microoperations) to control dataflow
* Microoperations – ops performed on data stored in registers, include register transfer, memory transfer, arithmetic microops, logic microops, and shift microops, take place in ALU
  + Control Function – P Boolean variable (function output) = 1 means enable microops, 0 means disable
* Register Transfer Language – transfers data between registers using microops
  + To access part of a register, use () after register name ex. A(L) or A(H)
* Little Endian (left) – group of bytes stored from first to last, big endian (right) is last to first
* Three-state gate – alternative to multiplexer
* Buffer – a device inserted between other devices to match impedance, prevent mixed interactions, and supply additional drive or relay capacity
* Adder and Subtractor can both use a single circuit
* Logic Microoperations:
  + Masking out – ANDing bits with 0 to clear them
  + Setting – ORing bits with 1
  + Complementing – XORing bits with 1
  + Clearing – ANDing bits with 1 then XOR bits with output (both are same)
* 3 types of Shift Microops: Logical, Arithmetic, and Circular
  + Serial Input – the ‘new’ bit of a shift
  + Logical – the serial input is 0
  + Arithmetic – serial input is 0 for shift right, is 1 for shift right
    - Used for division, multiplicaation
  + Circular – the serial input is the bit ‘knocked off’ the other side
* Arithmetic, logic, and shift circuits can be combined into Arithmetic Logic Unit (ALU)
* Memory Transfer Notation – gets value at the address of the address register (AR) with ‘M[AR]’
  + Memory Read and Write uses AR and data registers (DR): DR🡨M[AR], M[AR]🡪DR

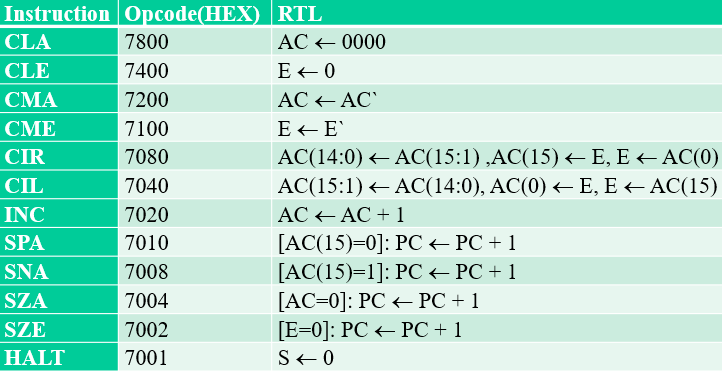
**CH 5 – Hardwired Control**

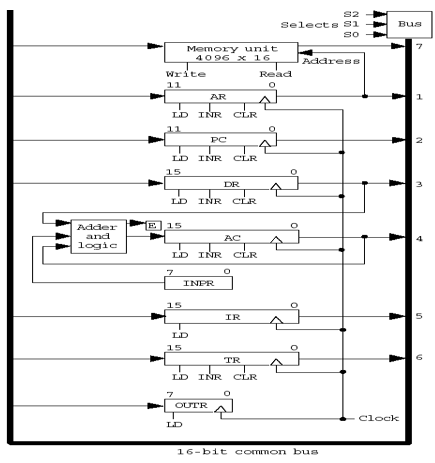
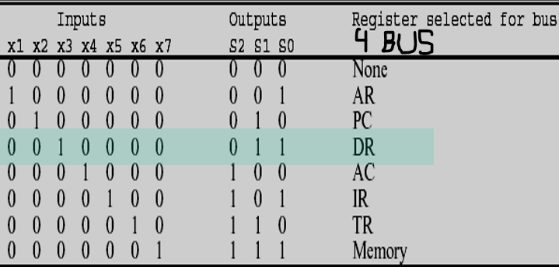
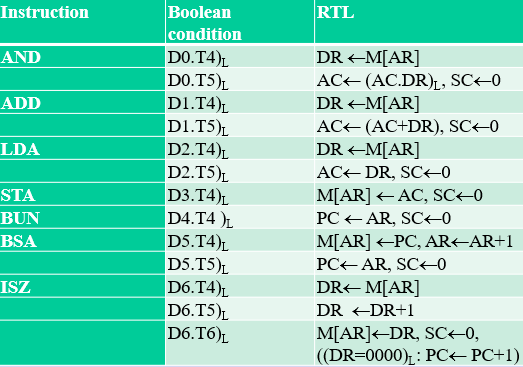
* Computer made up of Hardware (registers, buses, control, clock), Software (Assembly language, macro operations), and Register Transfer Language (Microoperations)
  + Control Unit – generates a time sequence of control signals to trigger microops
* Program statements/Instructions made of opcode and operands (data locations)
* Program – set of instructions that specify operations, data and control sequence
  + Interrupt Cycle – used to implement input and output
* Instruction – specifies a sequence of microoperations
  + Instruction Cycle – method of processing cycles
    - Fetch next instruction from memory, decode its value, execute microops
  + First bit of instruction code for addressing type: 0 is direct, 1 is indirect
  + Direct Addressing – when data part of an instruction is the data address
  + Indirect Addressing – when data part contains the address of another address, address where address is found
  + Instructions can have different formats:
    - Opcode + data (immediate operand)
    - Opcode + data address (direct addressing)
    - Opcode + address of address of data (indirect addressing)
    - Data only (no instruction)
    - Opcode only (no data needed)
* One Operand Instruction Code
  + 16 bit code, first 4 for instruction, then 12 for operand address
  + When input or output flag is 1, program is interrupted
* Registers of Mano’s Basic Computer:
  + When data is transferred from smaller to bigger registers, 0s are added, when vise versa, most significant bits are cut off
  + AR Register – Address Register, 3 bytes, gets the current instruction address from PC
  + IR Register – Instruction Register, 4 bytes, gets contents of AR register address from memory
  + DR Register – Data Register, 4 bytes, holds the operand(data) that is read from memory
  + AC Register – Accumulator Register, 4 bytes, general purpose processing register where most processing takes place, flip-flop E stores carry, not directly connected to the bus
  + TR Register – Temporary Register, 4 bytes, holds temp data during processing
  + INPR Register – Input Register, 2 bytes, holds input char, not connected to the bus, only ALU
  + OUTR Register – Output Register, 2 bytes, holds output character
  + PC Register – Program Counter, 3 bits, stores the address of the next instruction, is incremented to fetch next instruction
    - Instruction data is read and executed in sequence unless interrupted by a branch instruction
      * Branch instruction – leads to nonconsecutive instruction
  + ALU only takes input from DR, AC, and INPR
* Opcodes in Mano’s Simple Computer – first 4 bits of instruction word
  + 0111 is reserved for register reference instructions, 1111 reserved for I/O instructions, 14 other opcodes are memory reference instructions

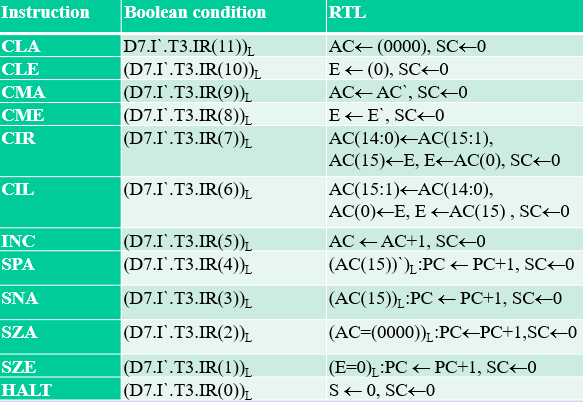


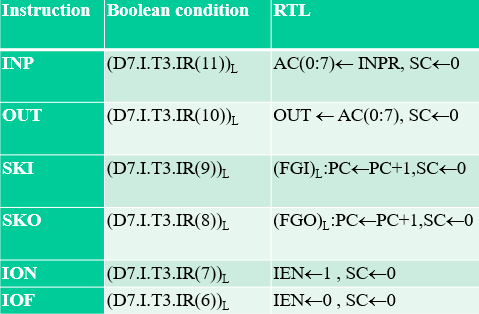


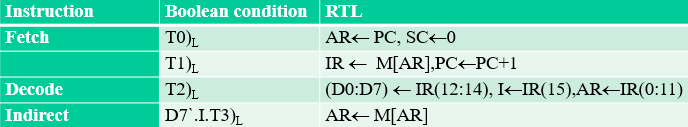


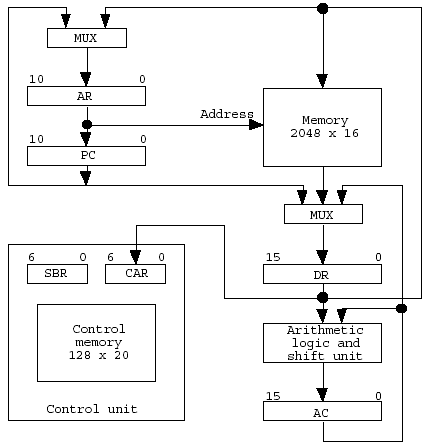


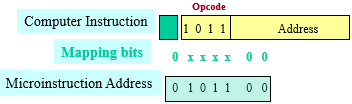
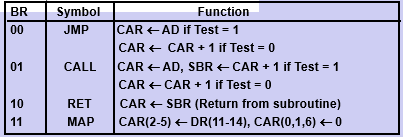
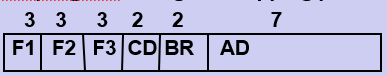
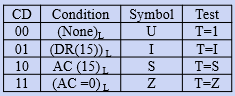
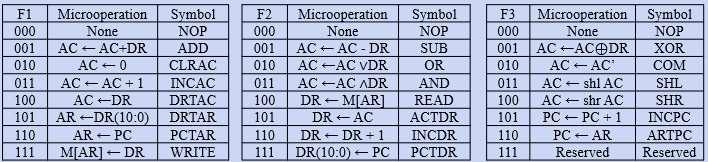
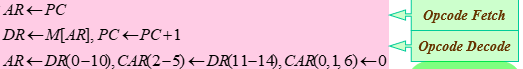
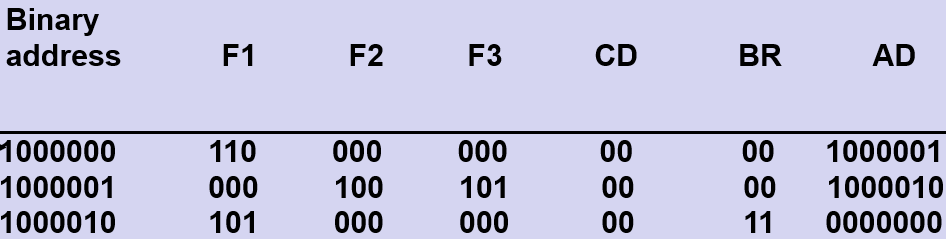
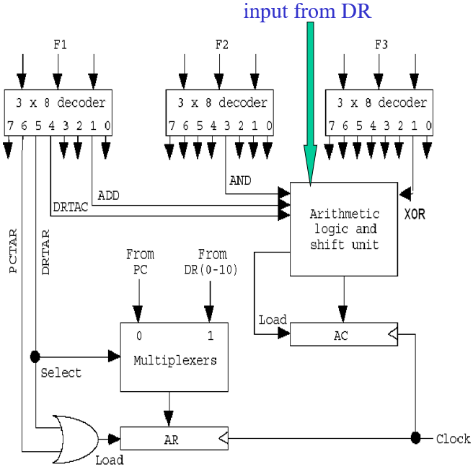
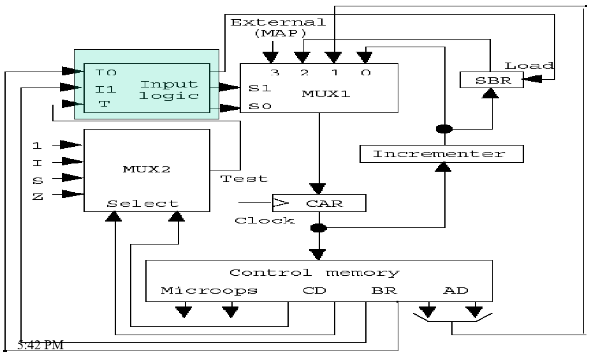
* Mano’s Basic Computer has 8 registers, a memory unit, and a control unit
  + Control Input: LD, INC, CLR, Write, Read
  + Paths must be provided to transfer info, bus system is most efficient
    - When the LD (load input) is enabled, that register receives the data from the bus
    - Bus is controlled by selection switches, they determine the destination of data
  + System Clock pulses define the shortest time interval betweem 2 successive actions
    - Only 1 timing signal can be set to 1 at once, all others in a cycle clock are set to 0
  + Control Unit – where all control signals (except clock) are generated including inputs for the bus, registers, and microops
    - Control Unit inputs come from instruction in IR
      * Flip-Flop used for the addressing mode bit, 3x8 decoder for 14-12 bits
* Instruction Cycle – method of processing a sequence of instructions
  + Sequence Counter – stores a 4 bit number and is incremented at each clock cycle
    - Sends number to a decoder which determines the timing signal T to set to 1
    - Can be cleared if SC 🡨 0, which causes T0 to be activated next clock cycle
  + 1: Fetch an instruction form memory (T0, T1)
  + 2: Decode the instruction (T2)
  + 3: Read effective address from memory if instruction has indrect address (T3)
    - Processed only if I bit is 1 and D7 = 0, if not, is skipped
  + 4: Execute the instruction , instruction table lookup (T4, T5)
  + Repeat until a Halt instruction is encountered (SC 🡨 0)







**CH 7 – Microprogrammed Control**

* Control unit decodes instruction into multiple control signals
  + Control signals can either be hardwired or microprogrammed
    - Hardwired control uses gates, flop-flops, decoders etc., are fast but inflexible, ch5 computer with bus
    - Microprogrammed control is flexible but slow due to memory access, virtual computer on right
* Microprogram in control memory performs microoperations to execute instructions
  + In dynamic microprogramming, the control memory is in the RAM
  + In static microprogramming, the control memory is in the ROM
* Control Unit is like its own minicomputer with CAR (address) and SBR (subroutine) registers, control memory, and sequencer (next address generator)
  + Control Memory are size 27 with 128 addresses, stores microprograms (4 addresses long)
  + CAR could receive input from a branch, mapping, subroutine return, or incrementing
    - A 4x1 multiplexer determines which is input
    - From mapping, the microinstruction address takes the 4 opcode bits from the instruction:
      * Microprograms have 4 addresses for microinstructions to use due to the last 2 bits after the opcode
        + If more microinstructions are needed, can use 1000000 to 1111111 space in control memory
    - Branch logic uses a 4x1 multiplexer for 4 different possibilities ^
* Subroutines – programs that are used by other routines, can be called anywhere from microprogram
  + Register stack used for branching off for subroutines
* Machine Instruction format for control unit instructions: 1 bit addressing type, 4 bit opcode, 11 bit address
* Microinstruction Format has 20 bits total made of up 3 microoperation fields, a condition field, branch field, and address field
  + Microinstructions have 3 microoperation fields: F0, F1, and F2
    - Each field has 3 bits, 7 possible microops for each
    - Symbolic form (symbol) is converted to RTL form (microoperation) with microprogram assembler
  + Condition (CD) field is condition for the branch
  + Branch (BR) field chooses address of next microinstruction with branch logic
  + Address (AD) field contains 7 bit branch address, if BR is 11, mapping is used
* Fetch Routine – fetches next instruction and decodes it
  + Stored on address 64 of control unit memory
  + RTL form to symbolic form to microop form:
* Microprogram ADD stored on address 0:
  + Opcode subroutines stored on addresses 0-63 (16 opcodes)
* Microinstruction Word fields are decoded with 3 3x8 decoders
  + For arithmetic microops, those specific decoder outputs go directly into the ALU as input
* Conditional Field (CD) bits are input for MUX2, which chooses which condition to test
  + MUX2 output and BR bots are input for input logic
* Input Logic output determines which address input MUX1 will take to transfer to CAR

**CH 8 CPU**

* CPU runs programs by fetching and decoding sequences of instructions, then executing them
* CPUs consist of a control unit, ALU and registers
* CPUs are organized into 3 categories:
  + Accumulator Organization: computation occurs in a single accumulator register
    - Ex. AC 🡨 AC + DR
  + General Register Organization: computation occurs in multiple general-purpose registers
    - Ex. R3 🡨 R1 + R2
  + Stack Organization: Computation occurs in a stack
    - Register stack is made up of multiple registers (stack addresses)
* With General Register Org, a control word has 14 bits in 3 parts:
  + 2 3 bit operand registers (SELA &SELB) a 3 bit destination register (SELD), and a 5 bit operation:
  + This selects the 2 register operands in the 2 multiplexers, the operation for them in the ALU, and the destination register in a decoder
  + 000 for any register selection signifies external input or output (not using a register)
* Stack organization – stores information on a stack, data is pushed and popped
  + Stack Pointer (SP) – a register that contains the address of the top item of the stack
    - In a 2x stack, the SP register has x bits, one combination for each address
  + Full Register – 1 bit register set to 1 when stack is full
  + Empty register – 1 bit register set to 1 when stack is empty
  + Data Register (DR) – holds data to be written into or read from the stack
* Different organizations have different instructions with different numbers of registers:
  + 0 address instruction – stack organization
  + 1 address instruction – AC register and memory, AC for result, accumulator organization
  + 2 address instruction – 2 registers or 2 memory locations are used, 1st operand for result
  + 3 address instruction – general register organization, 2 operand registers, 1 destination
* Elements that define a computer’s architecture: # address bits, word length, # registers, register rules, number of instructions, addressing modes, operand rules
  + Bus uses electrical signals to transfer data between system components
    - An electric signal for each address bit, data bit, and control signal bit
    - Control signals include # bits transferred, direction, and timing
    - When CPU needs to transfer to and from memory, it uses MAR register for memory address and MDR register for data, control signals control read/write
* Types of Addressing Modes:
  + Implied mode – operands are implied in instruction definition, stack system
  + Immediate mode – operand is in the instruction itself, 0 address instruction
  + Register mode – operands are in registers
  + Register indirect mode – registers store the address where the operands are
  + Autoincrement/autodecrement mode – same as register indirect mode but with added incrementing or decrementing of the register before or after each usage
  + Direct access mode – effective address of operand is in the address field of the instruction
  + Indirect address mode – instruction address mode gives address of the address where operand is stored
  + Relative address mode – PC is added to address field to get effective address [eax + 4]
  + Indexed addressing mode – contents of an index register is added to address field to get the effective address
  + Base register-addressing mode – contents of a base register is added to address field to get the effective address

**CH 9 Parallel Processing**

* Parallel processing is used to process data concurrently to cut execution time, increases throughput
  + Throughput – amount of processing that can be done during a time interval
  + 2 or more instructions can be processed at the same time
* 4 different processing types:
  + Single instruction stream, single data stream (SISD) – no parallelism
  + Single instruction stream, multiple data streams (SIMD)
  + Multiple instruction stream, single data stream (MISD)
  + Multiple instruction stream, multiple data stream (MIMD)

**CH 10 Computer Arithmetic**

Quiz 5

Pipeline - (n - k - 1) \* clock cycle

Nonpipeline – \* clock cycle