4 Computer Architecture

4.1 MARIE – A Simple CPU Model

MARIE Architecture

- 16 bits word
- 4-bit opcode & 12-bit address

Registers

Type		**Function**
Accumulator	**AC**	General-purpose
Memory Address Register	**MAR**	Stores memory address
Memory Buffer Register	**MBR**	Holds data read from to written to memory
Instruction Register	**IR**	Current instruction
Program Counter	**PC**	Address of next instruction

Register Transfer Language (RTL)

	Fetch	**Decode**	**Execute**
Load X	1. MAR ← PC 2. MBR ← M[MAR] 3. IR ← MBR 4. PC ← PC + 1	5. MAR ← X	6. MBR ← M[MAR] 7. AC ← MBR
Store X			6. MBR ← AC 7. M[MAR] ← MBR
Add X			6. MBR ← M[MAR] 7. AC ← AC **+** MBR
Subt X			6. MBR ← M[MAR] 7. AC ← AC **–** MBR
Jump X			6. PC ← MAR
JnS X			6. MBR ← PC 7. M[MAR] ← MBR 8. AC ← MAR 9. PC ← AC + 1

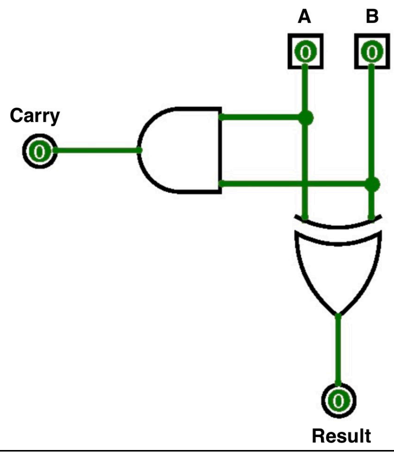
** Control Signals switch on / off based on RTL step **

4.2 Basic Circuits

Arithmetic Circuits
→ Perform arithmetic operations (+, -)

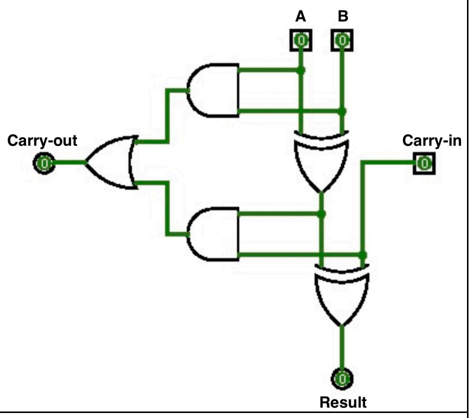
Circuit	Properties
Adders / Half Adders	• Adding two one-bit numbers

Input		Output	
A	B	Carry	Result
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1
Gates:		AND	XOR



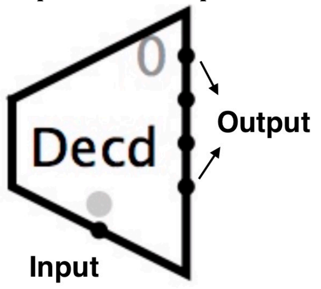
Circuit	Properties
Full Adders	• Adding three one-bit numbers

Input			Output	
A	B	C_{in}	C_{out}	Result
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1
Gates:			AND	XOR

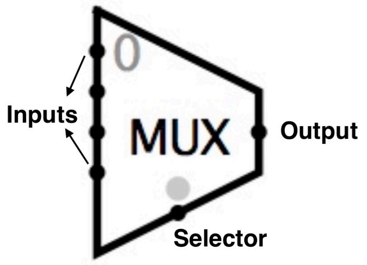


Circuit	Properties
Ripper -Carry Adders	• Combine full adders like a chain • Previous bit C_{out} = Next bit C_{in}

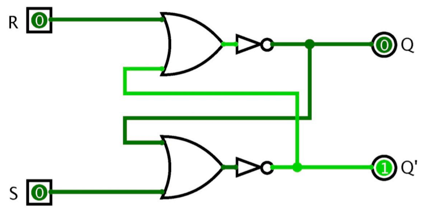
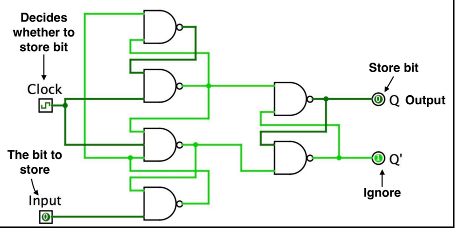
Conversion Circuits
→ Converts input to certain outputs

Circuit	Properties
Decoders	• Activates only one output based on binary inputs • n inputs & 2^n outputs 
Multiplexers	• Choose the data inputs to activate

Selector 0	Follow input 0
Selector 1	Follow input 1



Sequential Circuits
→ Outputs depend on sequence of inputs & outputs

Circuit	Properties
Flip Flop / S/R Latch (Set/Reset)	• Only one output can be activated at a time (loop constantly negates it) • Used to store single bit data 
D Flip Flop	• Similar to S/R latch, but with a "clock" that selects the bit to store  Decides whether to store bit Clock The bit to store Input Store bit Q Output Q' Ignore

4.3 CPU Construction

Components	Properties
Arithmetic Logic Unit (ALU)	• Implements basic computation: ⇒ Addition, subtraction, multiplication ⇒ Comparison ⇒ Boolean operations ⇒ Shifting (update variables)
Registers	• Collects all registers into a single circuit • Implemented using n flip-flops (for n bits) • With extra inputs(read & write)
Control Unit	• Perform fetch-decode-execute cycle through control signals • Function of control signals: ⇒ Read/write register ⇒ Read/write memory address ⇒ Perform operations in ALU **refer to RTL section above**

4.4 Memory

Memory Addressing

Type	Per memory location
Byte-addressable	One byte [8 bits]
Word-addressable	One word [16 bits]

RAM

- Made up of multiple chips
- Each chip has fixed size of L x W:

⇒ L: number of locations

⇒ W: number of bits per location

- 2K x 8:
  = 2 x 10^{10} x 2^3
  = 2^{14} bits per chips

RAM Addressing

- RAM of 2^{17} locations
- Each address 17 bits long

010	110	00010010011
Row 2	Column 6	Byte 147

- Use MUX to select the chips

4.5 Input / Output Devices

- Each I/O device has its own registers

I/O Access Method

	Memory-mapped	Instructions-based
Address Space	Shared between memory and I/O	Individual for memory and I/O
Access	Regular instructions (MARIE)	Special instructions (In and Out)
Pros	• No new instructions • Simple	• Less logic to decode I/O addresses
Cons	• I/O used up memory • RAM available reduced	• More instructions (send data & perform)

I/O Perform

	Programmed	Interrupts
Function	Checks I/O at regular intervals	Notifies CPU for I/O requests & execute Interrupt Handlers
Pros	• Simple • Can control frequency of checking	• Fast
Cons	• Must check periodically • Slow • Increased power usage	• Hard to work with different priorities (graphics card & mouse)

Interrupt Parts

Parts	Properties
Interrupt Signals	• Notifies CPU and sets a bit in special register • Use fetch-decode-execute cycle • If bit set, call Handler
Interrupt Handlers	• Must leave CPU in the same state as before • Run by context switch ⇒ Via shadow registers – CPU switches to separate registers ⇒ Via programming – save previous location to memory
Interrupt Vectors	• Individual identification number for each device • CPU stores that number in special register • Handlers use it to jump to interrupt vectors (a list of subroutines)

Direct Memory Access (DMA)

- Improved version for interrupts
- CPU dedicate memory transfer to special controller
- CPU and DMA share same data bus (operate one at a time)