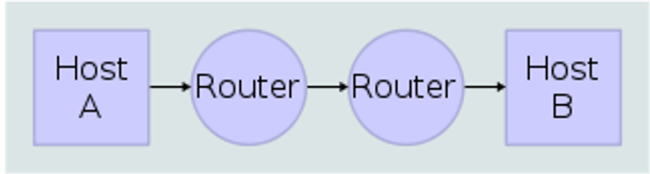
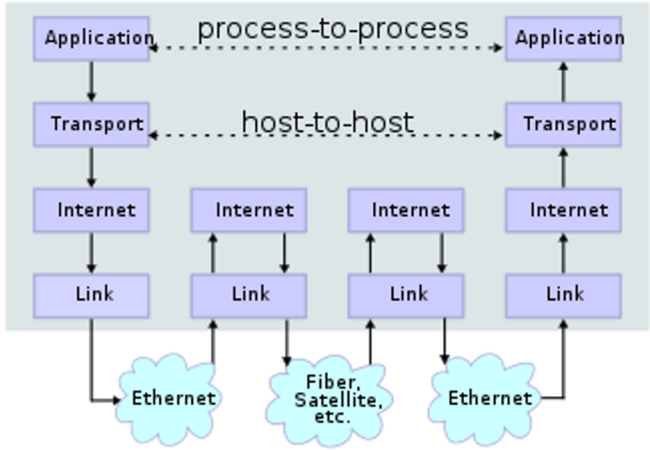
Models of Computation and Communication

- We need to develop a model
  - to reason about the problem
  - to reason about our solution
  - to reason about the problem about our solution.
- Models of Communication:
  - OSI/ISO model (Open Systems Interconnect)
  - TCP/IP model
- Model of Computation: (Machine <-> Language)
  - Turing Machine, Linear Bounded Automata, Pushdown Automata, and Finite State Automata
  - Sequential Circuits, and Combinational Logic
  - ➢ Universal Computer and Machines: Theoretical to Abstract to Physical

Network Topology



Data Flow

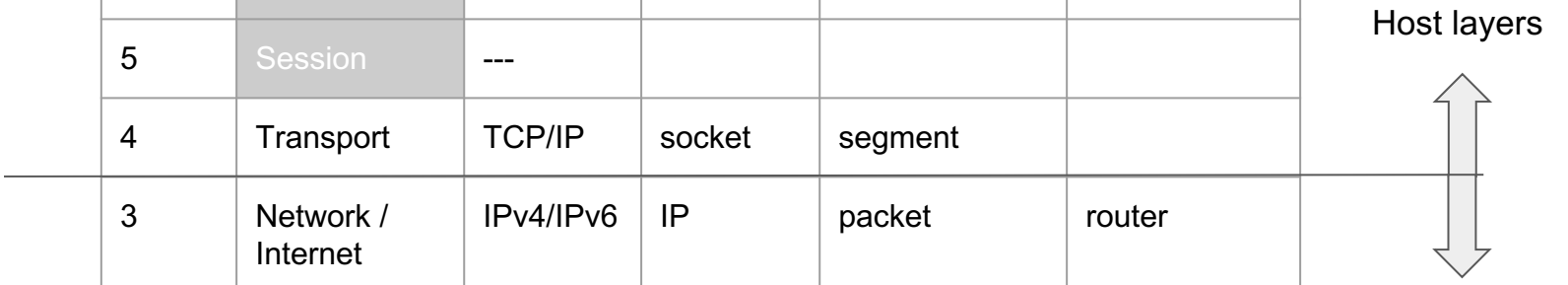


IPv4 Packet Header

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Version				IHL				DSCP						ECN		Total Length															
4	32	Identification																Flags			Fragment Offset											
8	64	Time To Live								Protocol								Header Checksum															
12	96	Source IP Address																															
16	128	Destination IP Address																															
20	160	Options (if IHL > 5)																															
⋮	⋮																																
60	480																																

OSI and TCP/IP Models

Layer	Name	Example Protocol	Naming	Transported	Hardware Device
7	Application	http	url	data	
6	Presentation	---			
5	Session	---			
4	Transport	TCP/IP	socket	segment	
3	Network / Internet	IPv4/IPv6	IP	packet	router
2	Data Link / Link	Ethernet	MAC	frame	switch
1	Physical	802.11g	Interface	symbols	hub, bridge



The Layers Simplified

Layer 1: Physical Layer

- The mechanics of sending symbols -- restricted (maybe) to one's and zero's

Layer 2: Data Link

- When to start and stop an individual message between two connected location

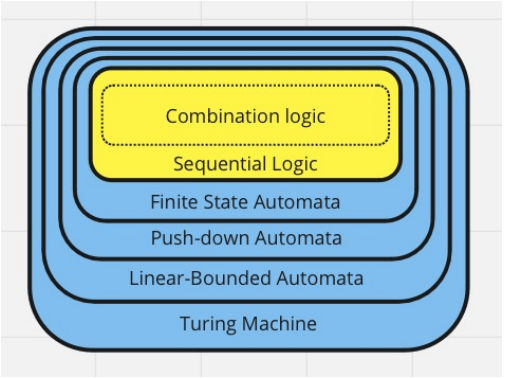
Layer 3: Network

- Sending a message from A ⇒ Z by going through B to C to D to … to Y and then finally Z

Layer 4: Transport

- Transmitting/Ensuring a complete message from A to Z
- Address performance issues

Models of Computation



Turning Machines	Recursively Enumerable	TM(Q, Σ, **Γ**, q0, **δ**)
Linear Bounded Automata	Context Sensitive Languages	LBA(Q, Σ, **Γ**, q0, **δ**)
Pushdown Automata	Context Free Languages	PDA(Q, Σ, **Γ**, **δ**, q0, z0, F)
Finite State Automata	Regular Expressions	FA(Q, Σ, **δ**, q0, F)
Sequential Circuits		
Combinational Logic	Boolean Algebra	

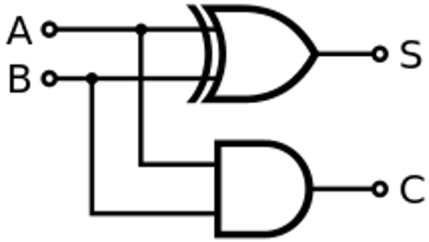
Combinational Logic

java: A || B

A	B	⊕
False	False	False
False	True	True
True	False	True
True	True	False

- Based upon Boolean Algebra
  - all inputs and outputs restricted to True (1) and False (0)
- Operations are restricted to: AND (*) , OR (+), NOT (')
- Equivalent to Digital Logic, with gates:

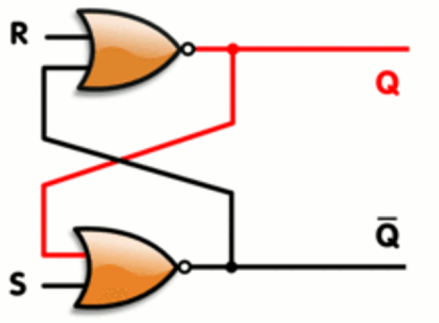
- Can be used as a building blocks: 
  - XOR: A ⊕ B is equivalent to (A + B) * (A' + B')
- Example: Half-Adder



Sequential Circuits

- Introduce feedback loops
- Creates latch or flip-flop
  - a circuit with only two stable states
- Example: SR Latch

S	R	Q	Output	Description
0	0	Q	Q	Hold State
0	1	Q	0	Reset / Clear
1	0	Q	1	Set
1	1	Q	X	Not allowed: Error

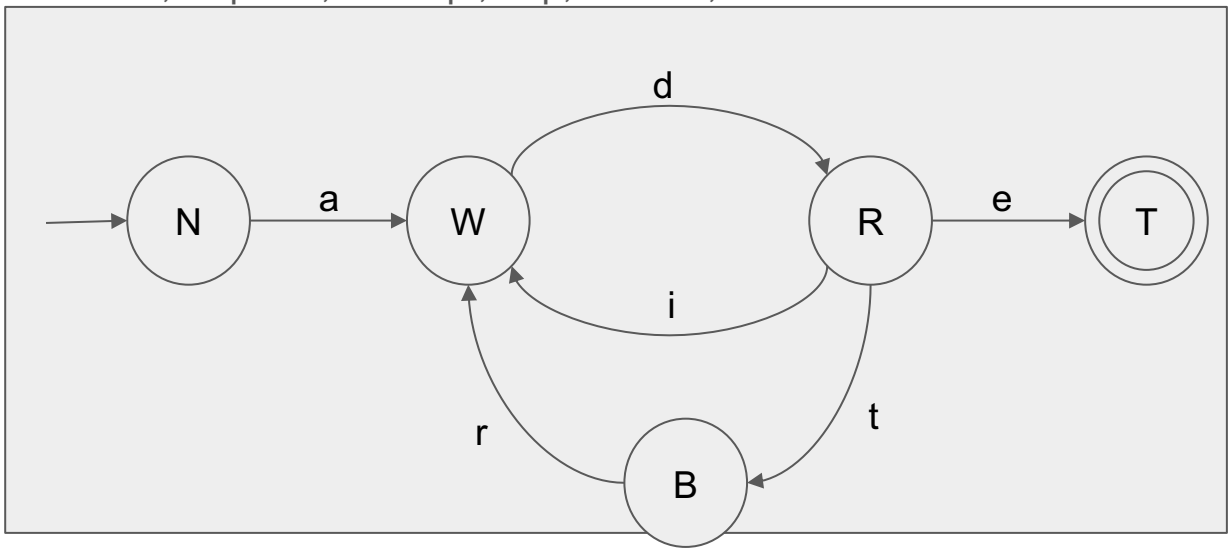


Finite State Machine

- FA(Q, Σ, **δ**, q0, F)
  - Q = { N, W, R, B, T } // New, Waiting (Ready), Running, Blocked, Terminated
  - Σ = { a, d, i, t, r, e} // admit, dispatch, interrupt, trap, resume, exit
  - q0 : N
  - F : { T }
  - **δ** : Q x Σ -> Q

input string:

a	d	t	r	d	a	d	e	$



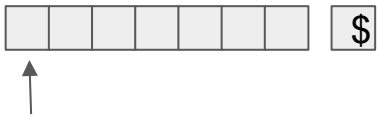
FA for the Process Status Diagram

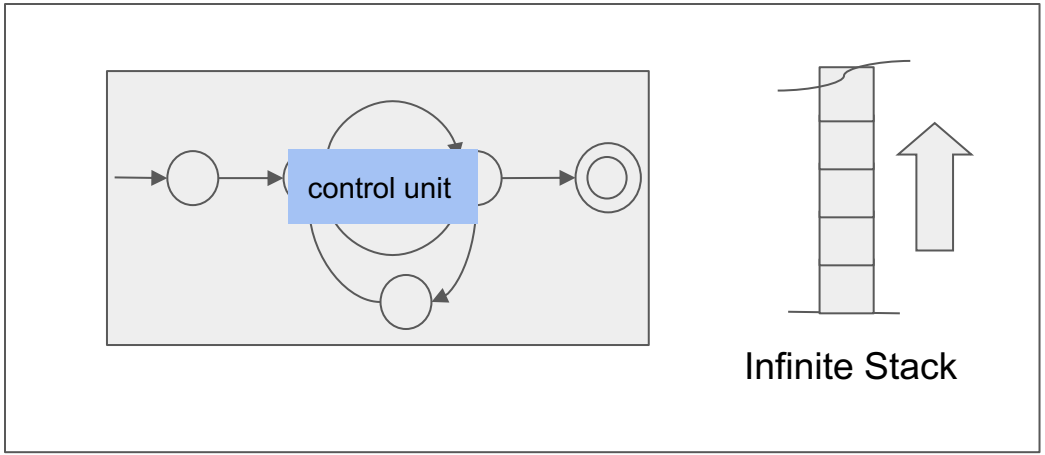
Pushdown Automata

```
C -> if ( E ) S
   |  if ( E ) S else S
```

- PDA(Q, Σ, **Γ**, **δ**, q0, z0, F)
  - Σ : set of symbols on the input string
  - **Γ** : set of symbols placed on the stack
  - z0: set of symbols place on the stack at startup
  - **δ** : Q x Σ x **Γ**-> Q x **Γ***

input string:



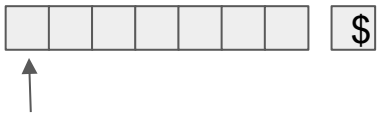


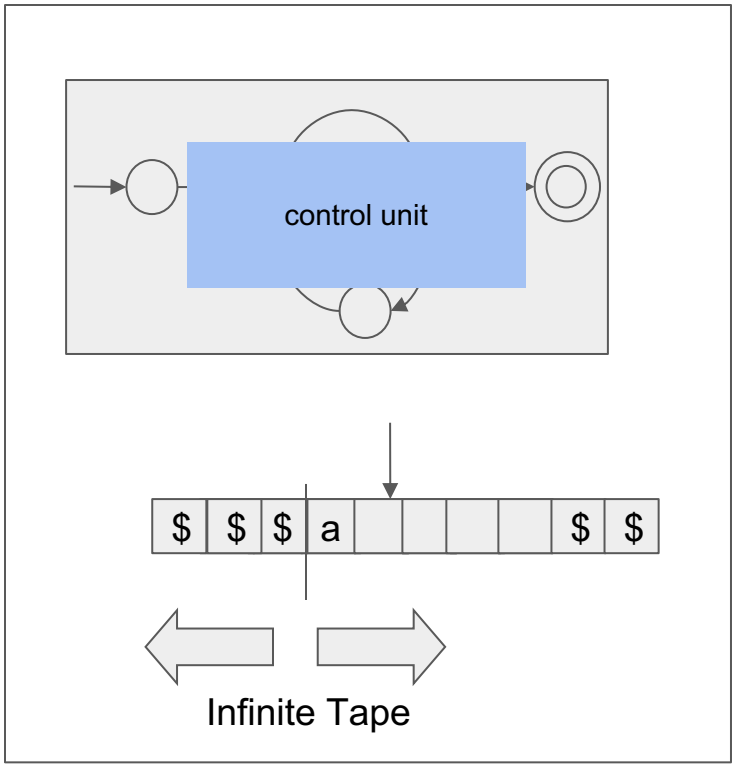
PDA

Turing Machine

- TM(Q, Σ, **Γ**, **δ**, q0)
  - Σ : set of symbols on the input string
  - **Γ** : set of symbols placed on the tape
    - includes a blank symbol: $
  - **δ** : Q x Σ x **Γ**-> Q x **Γ** x {R, L}

input string:



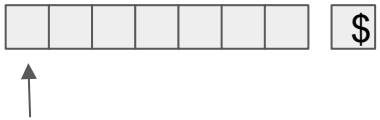


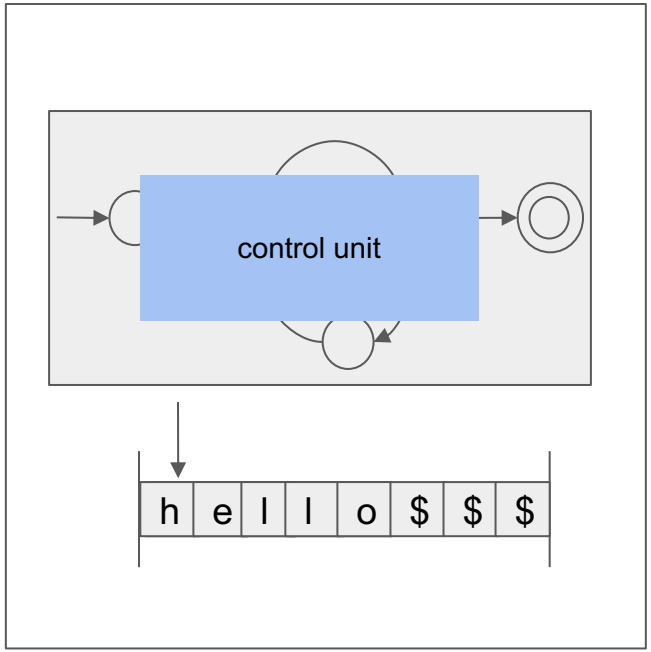
Turing Machine

Linear Bounded Automata

- Special Case of a Turing Machine
  - The tape is bounded to a defined size.

input string:





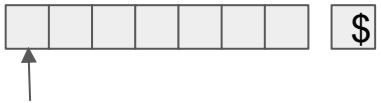
LBA with tape size of 8

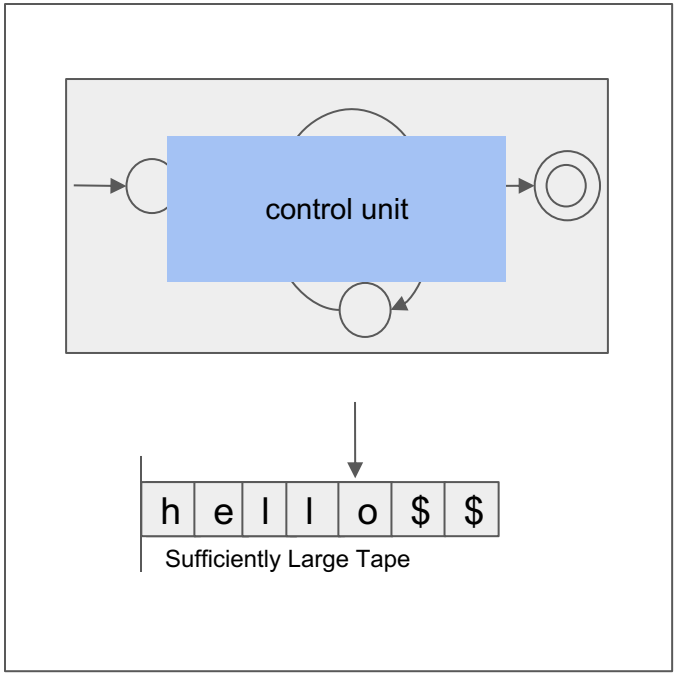
Universal Computer

- TM(Q, Σ, **Γ**, **δ**, q0)

- Tape: sufficiently large
- A specialized control unit (aka firmware)
- A specialized program placed on tape
- A generic program placed on tape
- Input coming from an I/O device

input string:

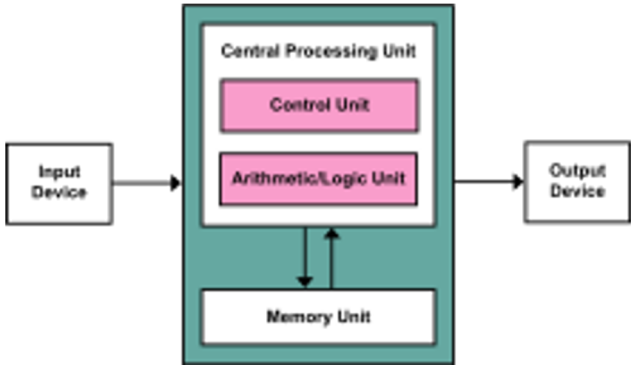


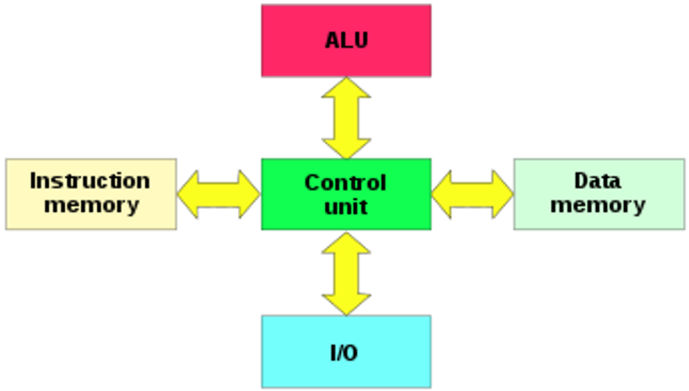


Universal Computer

Theoretical to the Abstract

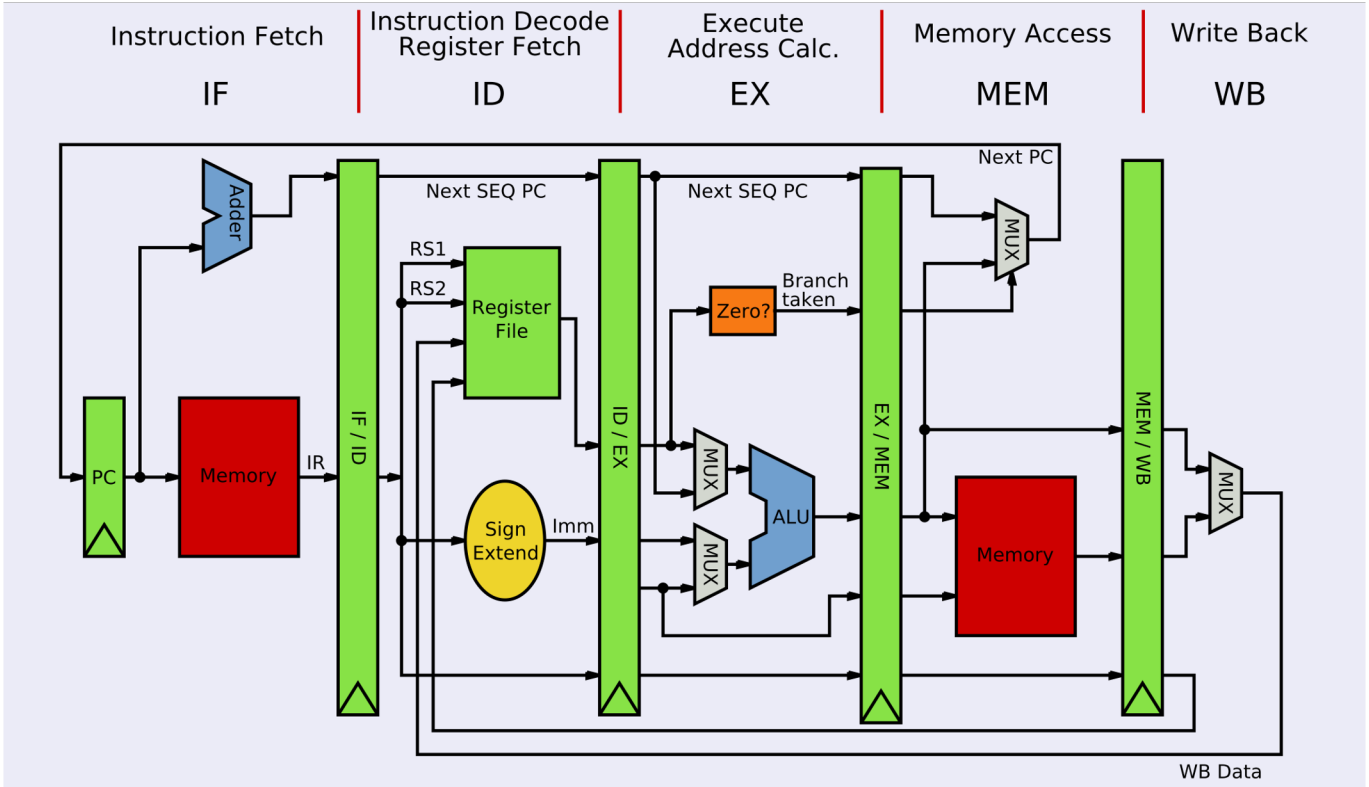
- Turing Machine →
  - von Neumann Architecture
  - Harvard Architecture



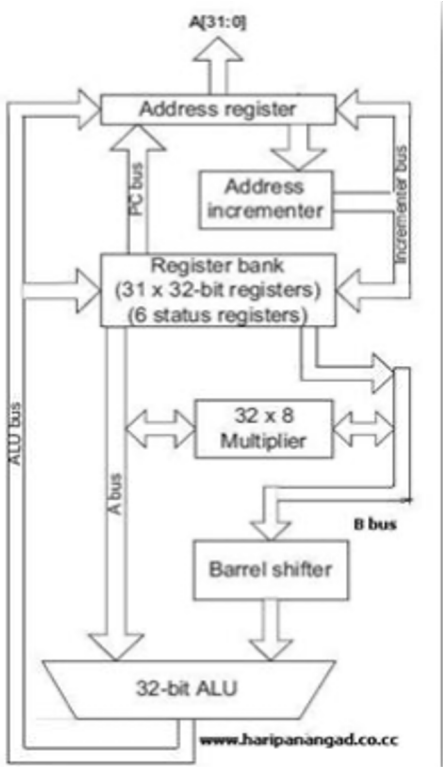


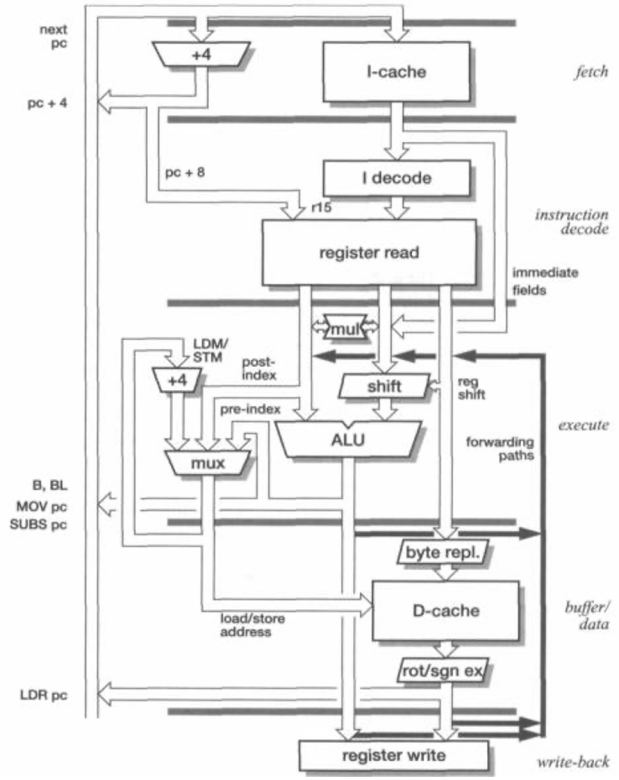
- Consider writing a Java program for these machines

Physical Architecture: MIPS Microarchitecture



Physical Architecture: ARM (7&9) Microarchitectures





Physical Architectures: Examples