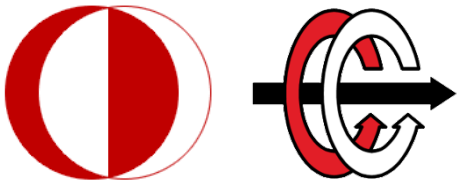


EE 445 Computer Architecture I

Fall 2025

Administrative Details-Instructors

- Section 1: Ece Güran Schmidt
  - Email: eguran@metu.edu.tr
  - Web page: http://users.metu.edu.tr/eguran/
  - Office: A402
- Section 2: Ahmed Hareedy
  - Email: ahareedy@metu.edu.tr
  - Web page: http://users.metu.edu.tr/ahareedy/
  - Office: D124-2
- You can check out our web pages to see what we do in our research

Administrative Details

- Schedule
  - Section 1: Tuesday 14:40-15:30 Thursday 10:40-12:30, @ A209
  - Section 2: Tuesday15:40-17:30 Thursday 16:40-17:30, @ A206
- Course Conduct
  - In-class lectures and problem solving sessions. Video lectures of the previous years (subject to change) and lecture notes are posted in odtuclass.
- Required
  - Prerequisite(s): EE348
  - Core course for Computer Option

Grading

- 4 Short Exams: 52% (13% each)
- Final exam: 36%
- HDL Homeworks: 12%
- 5% bonus for attendance >=80%

Schedule

Week	Topics
1	Introduction and 348, ASM
2	ASM
3	ASM, RTL
4	Basic Computer
5	HDL Lecture
6	Basic Computer
7	Basic Computer, Question Solving
8	Microprogramming
9	Microprogramming, Question Solving
10	Arithmetic Processor
11	Arithmetic Processor, Question Solving
12	Floating Point, ARM ISA
13	ARM ISA
14	ARM ISA, Question Solving

Administrative Details

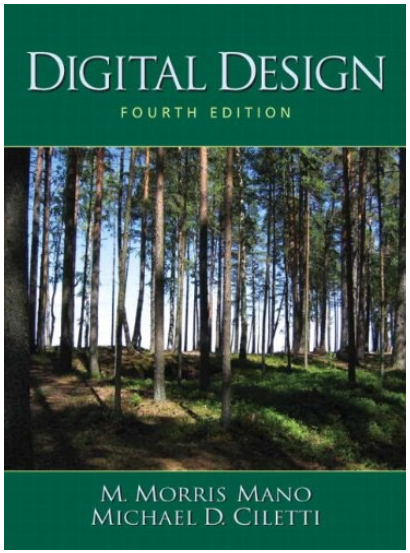
- **Follow**
  - https://odtuclass.metu.edu.tr/

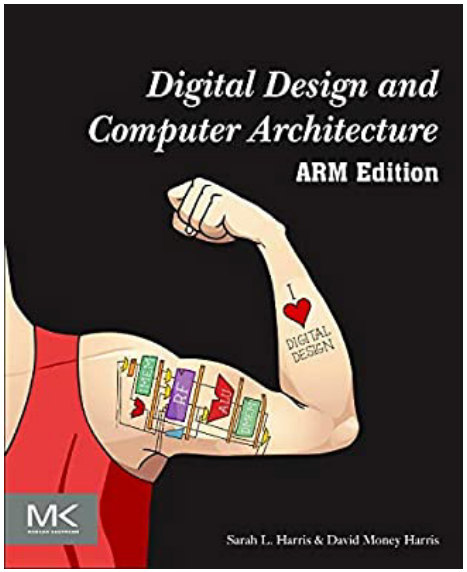
  for lecture slides, all class material, recorded lecture videos and announcements
  - Your e123456@metu.edu.tr email
- **Communication**
  - Preferred communication mean: E-MAIL
  - Send with subject **including** EE445 (no guarantee of reply otherwise)

Text Books

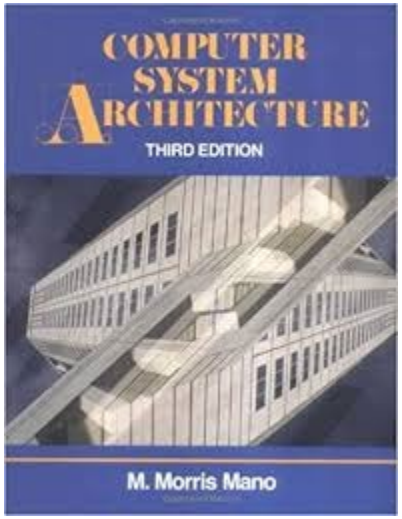
Copyright notice: Lecture Note Slides are compiled from the teaching material of these books, previous lecture notes of EE445 and additional resources. Part of the slides are entirely created by the instructors.

Digital Design (4th Edition)
M. Morris Mano, Michael D. Ciletti
Published by Prentice Hall, 2006





Computer System Architecture 3rd Ed.,
M. Morris Mano Prentice Hall, 1992
Computer System Architecture 2nd Ed.,
M. Morris Mano Prentice Hall, 1982



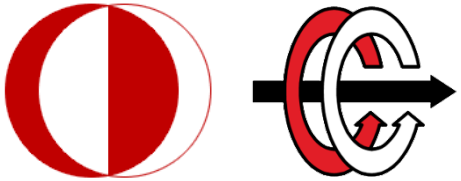
Harris & Harris, "Digital Design and Computer Architecture. ARM Edition", 1st Ed., Kaufmann, 2015.
We go on with this text book for EE446

Course Objective (Why should you take this course?)

- A smooth extension of EE348
- Describes how a computer works at EE348 level of detail on a simple fictitious Basic Computer
- Preparation for the advanced topics: pipelining, memory and I/O organization that are covered in EE446

Course Outline

- Introduction to Computer Architecture
- EE348 review
- Algorithmic State Machine
- Register Transfer Language
- HDL
- Basic Computer Architecture
- Computer Organization and Microprogramming
- Arithmetic Processor Design
- ARM Instruction Set Architecture

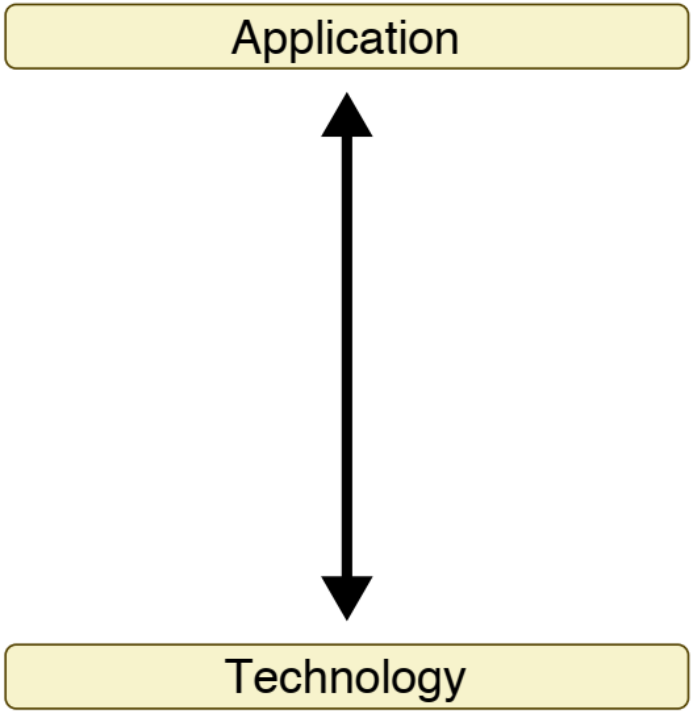


Introduction to Computer Architecture

Resources:

http://www.csl.cornell.edu/courses/ece4750
https://safari.ethz.ch/digitaltechnik/spring2020/doku.php
Computer Architecture
A Quantitative Approach, Sixth Edition

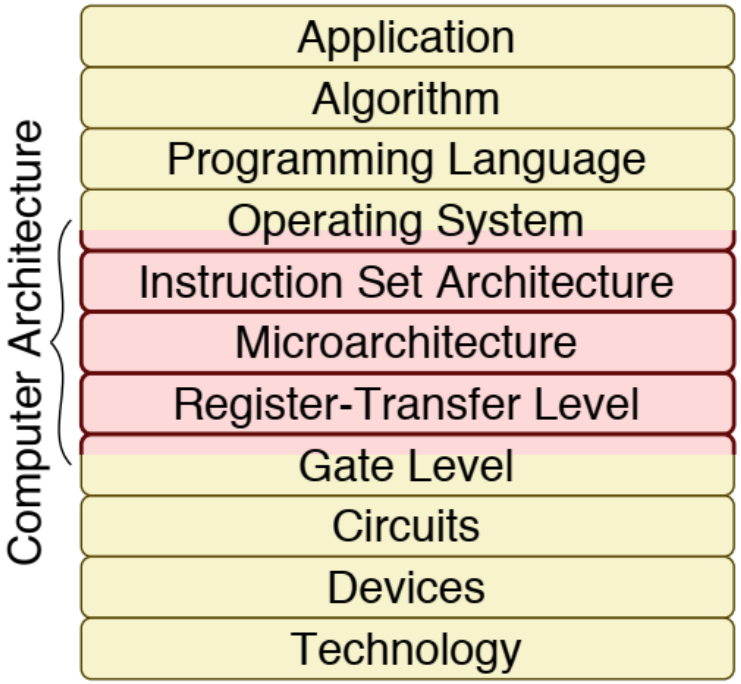
Computer Architecture



- abstraction/implementation layers
  - to execute information processing applications
  - efficiently using available manufacturing technologies

https://www.csl.cornell.edu/courses/ece4750/handouts/ece4750_overview.pdf

Computer Architecture



https://www.csl.cornell.edu/courses/ece4750/handouts/ece4750_overview.pdf

Sort an array of numbers

2,6,3,8,4,5 -> 2,3,4,5,6,8

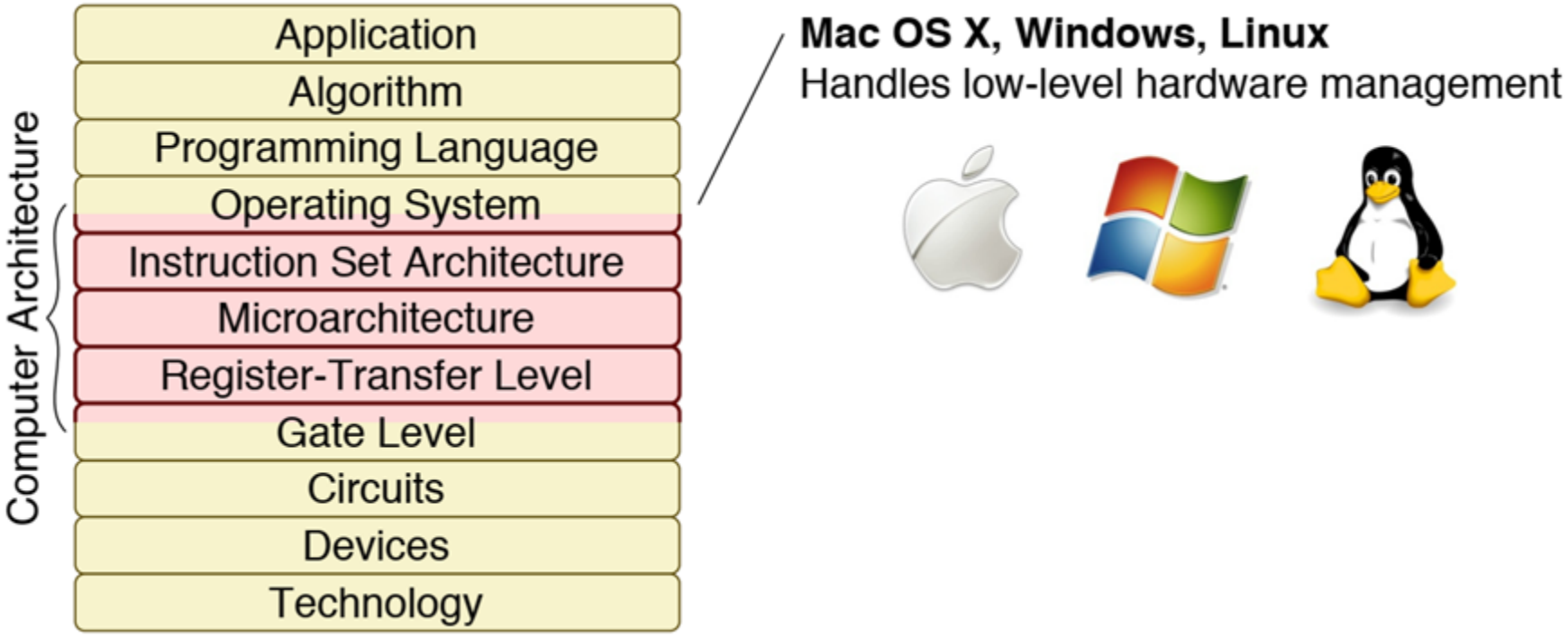
Out-of-place selection sort algorithm

1. Find minimum number in array
2. Move minimum number into output array
3. Repeat steps 1 and 2 until finished

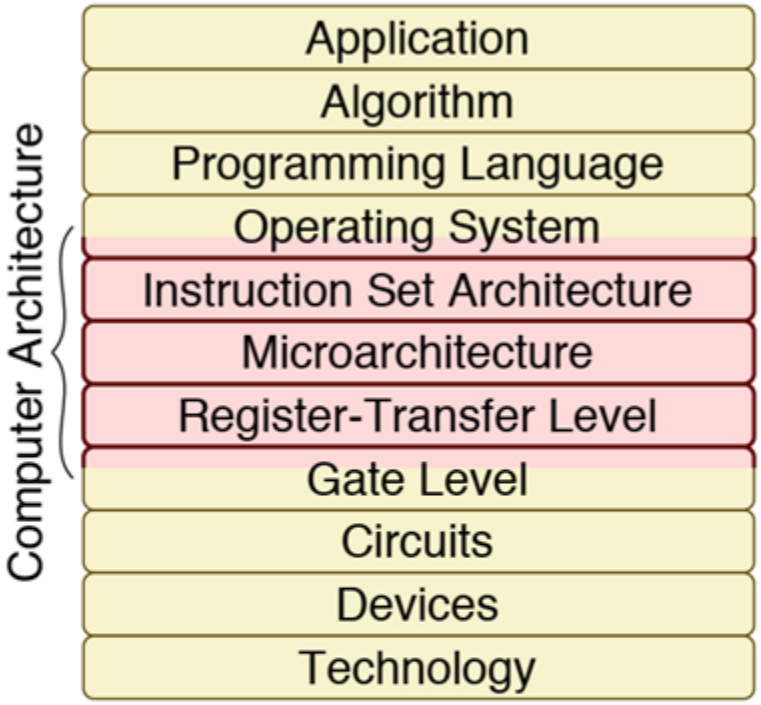
C implementation of selection sort

```
void sort( int b[], int a[], int n ) {
  for ( int idx, k = 0; k < n; k++ ) {
    int min = 100;
    for ( int i = 0; i < n; i++ ) {
      if ( a[i] < min ) {
        min = a[i];
        idx = i;
      }
    }
    b[k]   = min;
    a[idx] = 100;
  }
}
```

Computer Architecture



Computer Architecture



- Instruction Set Architecture (ISA):
  - Structure and behavior of the computer as seen by the programmer
  - There can be many implementations of the same ISA

MIPS32 Instruction Set

Instructions that machine executes

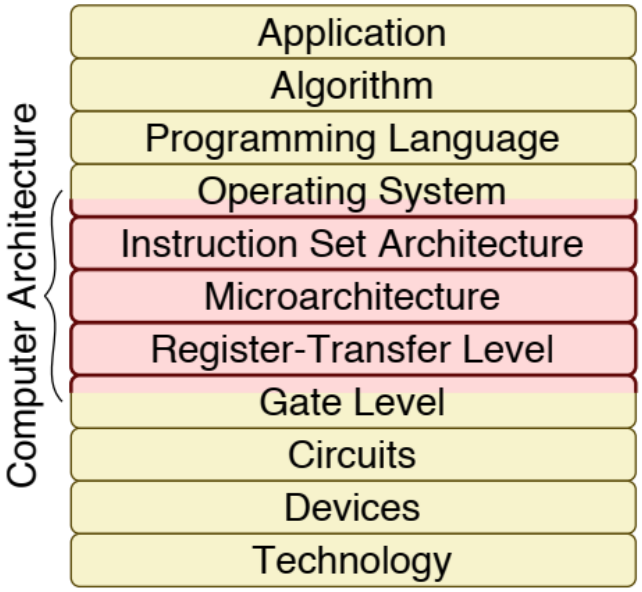
```
blez  $a2, done
move  $a7, $zero
li    $t4, 99
move  $a4, $a1
move  $v1, $zero
li    $a3, 99
lw    $a5, 0($a4)
addiu $a4, $a4, 4
slt   $a6, $a5, $a3
movn  $v0, $v1, $a6
addiu $v1, $v1, 1
movn  $a3, $a5, $a6
```

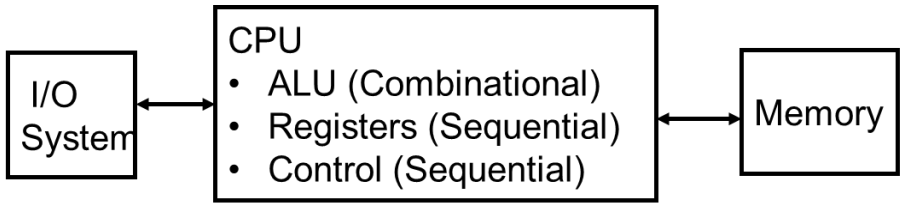
Instruction Set Architecture (ISA)

- Represents
  - all the information necessary to write a machine language program that will run correctly on the machine
  - the conceptual structure and functional behavior
- Abstracts away
  - the organization of the data flows and controls
  - the logic design
  - the physical implementation.
- Enables implementations of varying cost and performance to run identical software
- Includes
  - Addressing modes
  - Operand specifications
  - Operation specifications
  - Control flow instructions

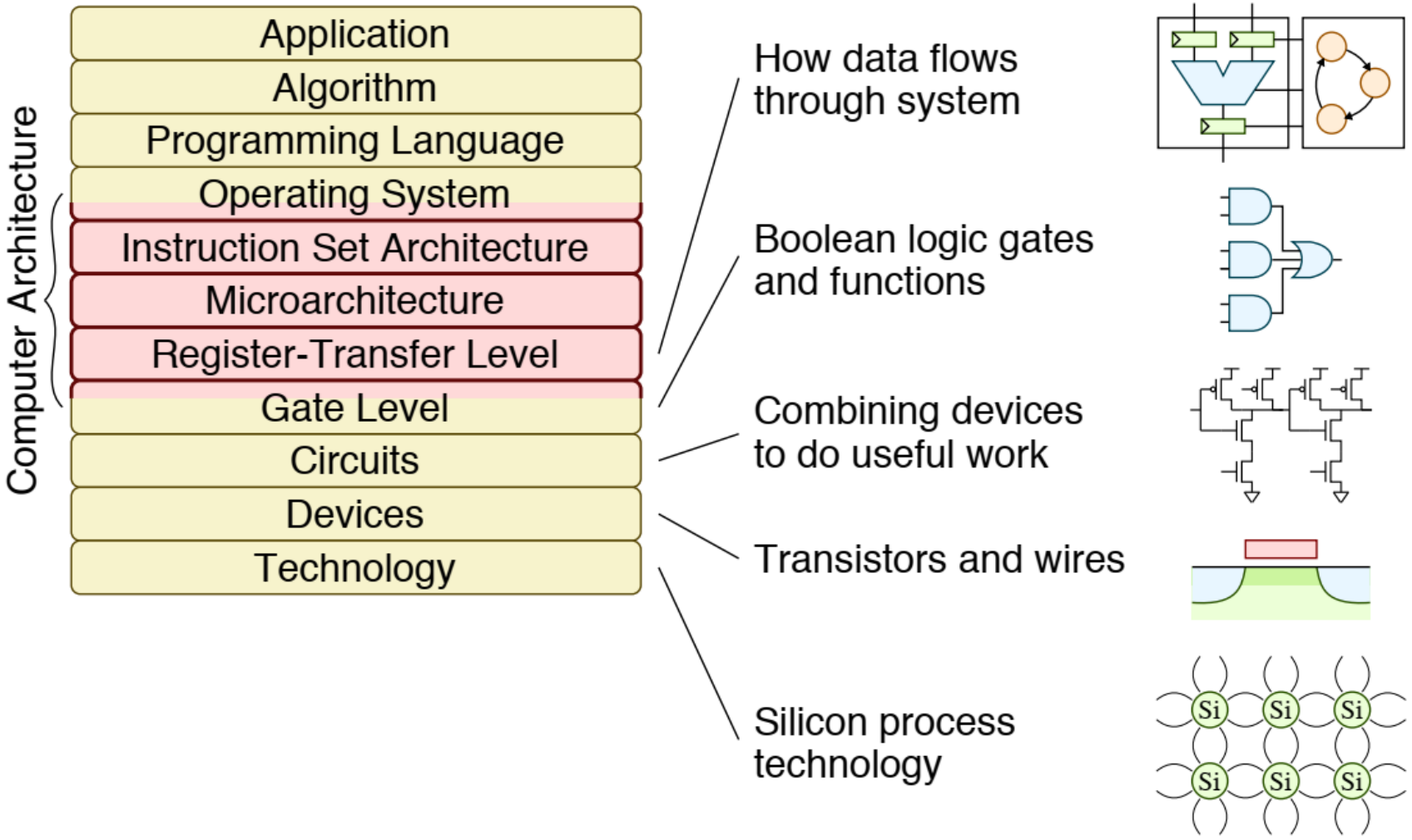
Microarchitecture

- Microarchitecture/Organization: The specific arrangement of registers, ALUs, finite state machines (FSMs), memories, and other logic building blocks needed to implement an ISA.
- Example: AMD Opteron and the Intel Core i7 implement the 80x86 instruction set with very different pipeline and cache organizations

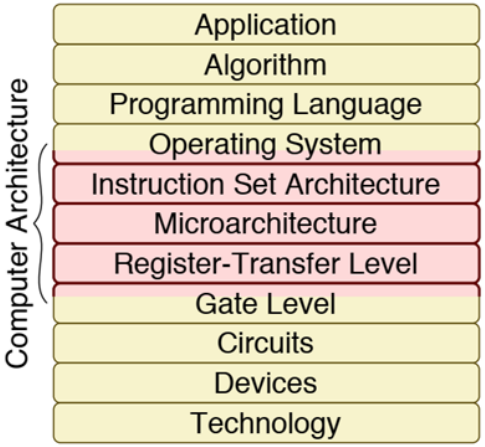




Computer Architecture



EE445 Coverage

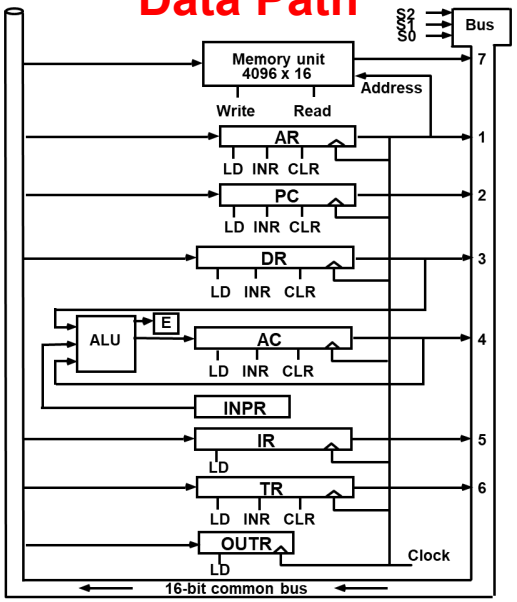


EE445 focuses on these layers using a fictitious Basic Computer

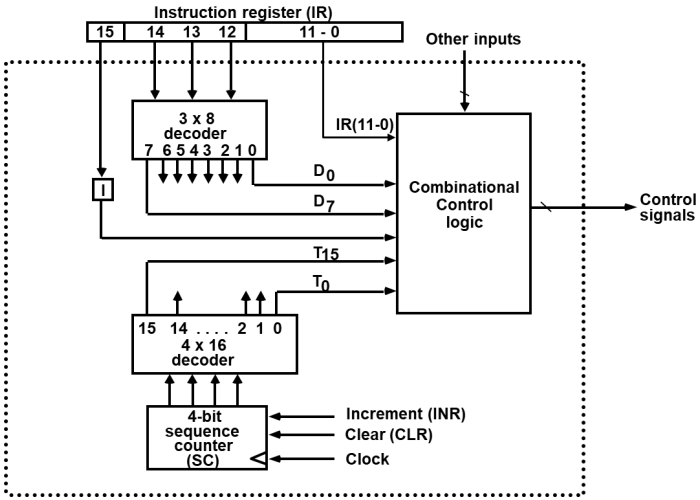
Instruction Set

Symbol	Hex Code I = 0	Hex Code I = 1	Description
AND	0xxx	8xxx	AND memory word to AC
ADD	1xxx	9xxx	Add memory word to AC
LDA	2xxx	Axxx	Load AC from memory
STA	3xxx	Bxxx	Store content of AC into memory
BUN	4xxx	Cxxx	Branch unconditionally
BSA	5xxx	Dxxx	Branch and save return address
ISZ	6xxx	Exxx	Increment and skip if zero
CLA	7800		Clear AC
CLE	7400		Clear E
CMA	7200		Complement AC
CME	7100		Complement E
CIR	7080		Circulate right AC and E
CIL	7040		Circulate left AC and E
INC	7020		Increment AC
SPA	7010		Skip next instr. if AC is positive
SNA	7008		Skip next instr. if AC is negative
SZA	7004		Skip next instr. if AC is zero
SZE	7002		Skip next instr. if E is zero
HLT	7001		Halt computer
INP	F800		Input character to AC
OUT	F400		Output character from AC
SKI	F200		Skip on input flag
SKO	F100		Skip on output flag
ION	F080		Interrupt on
IOF	F040		Interrupt off

Data Path

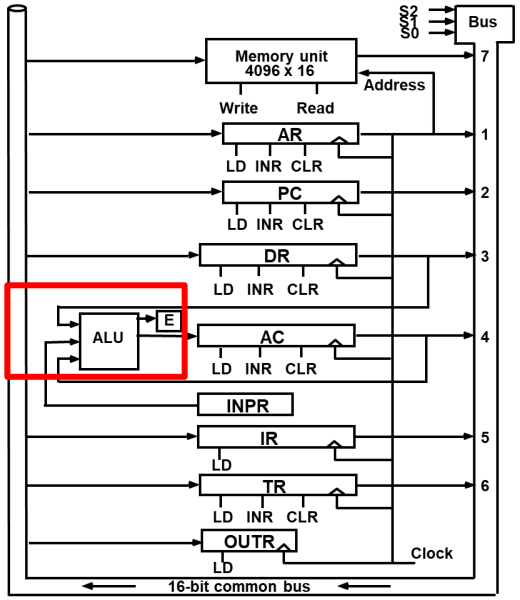


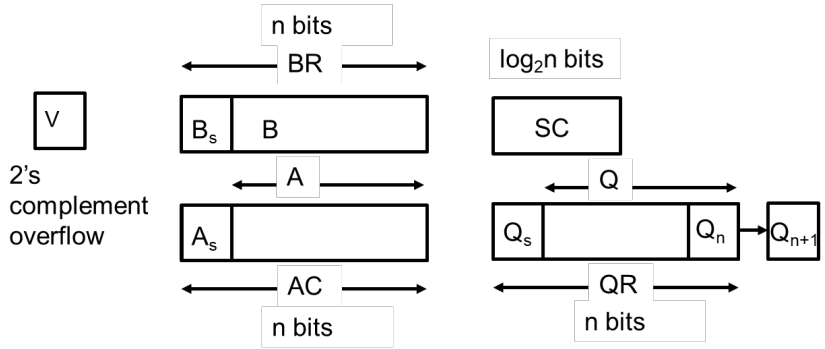
Controller



EE445 Coverage

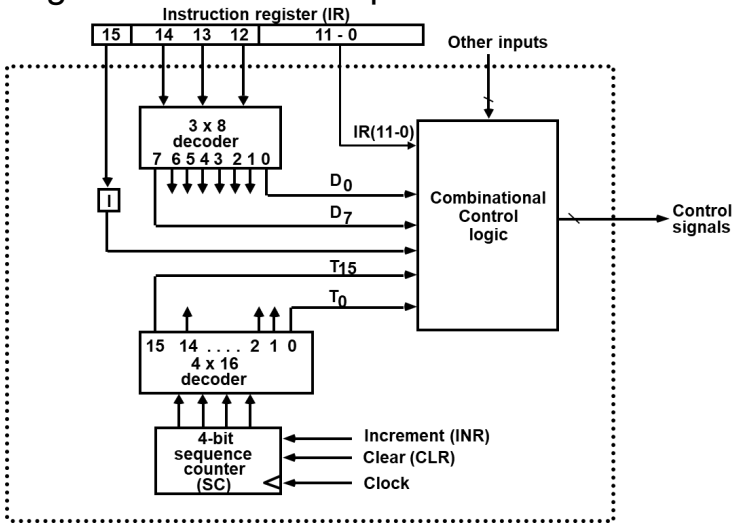
Data Path





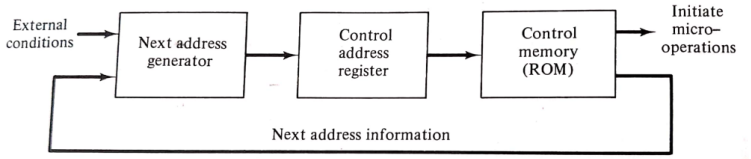
Hardwired control

Control signals are circuit outputs



Microprogrammed Control

Control signals are the control memory word contents



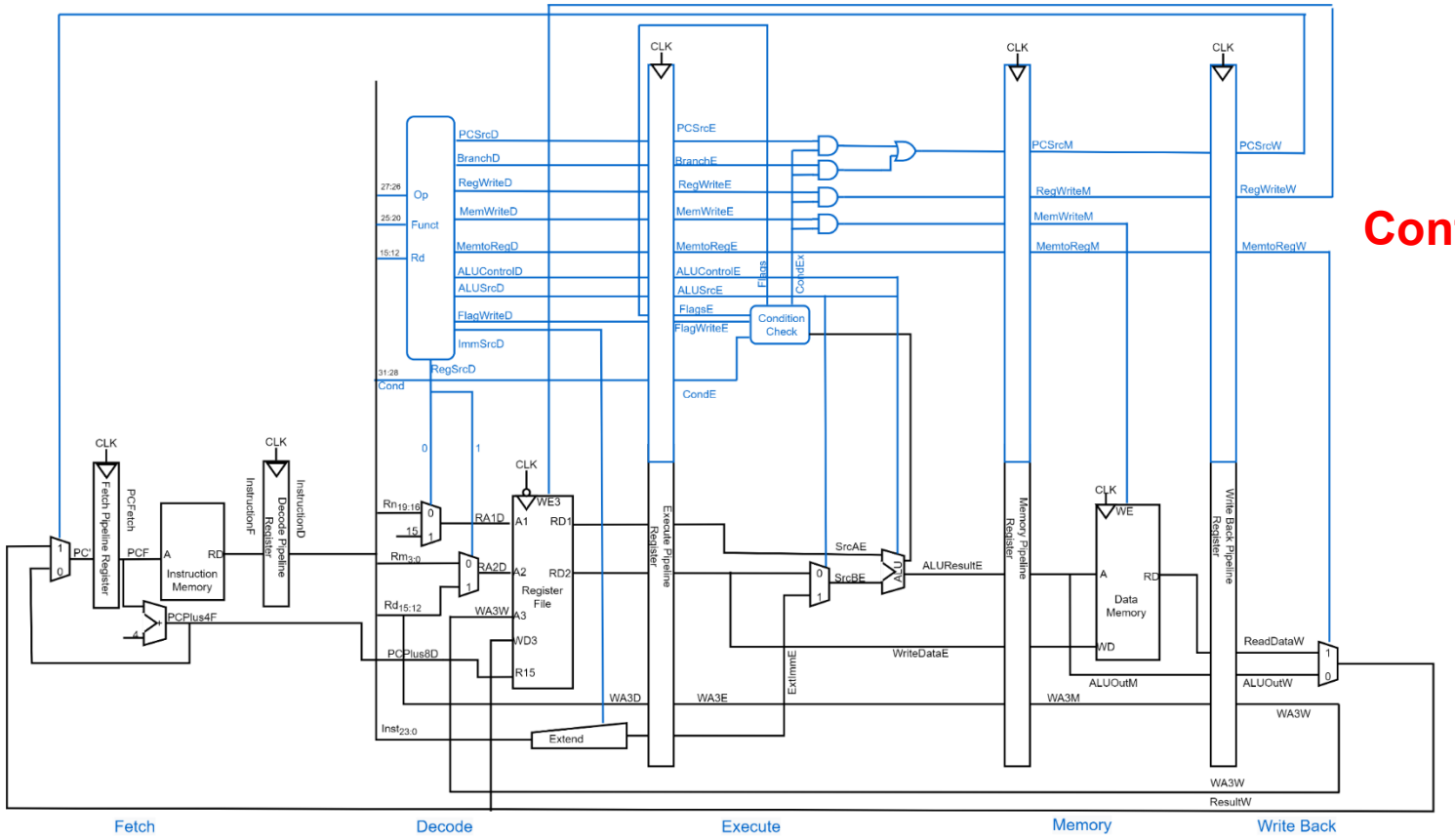
ALU implementation, hardware algorithms for multiplication and division➔ Needs EE348 refresher ☺

A Sneak Peek into EE446 ☺

A Pipelined Microarchitecture for a *representative* subset of ARM ISA

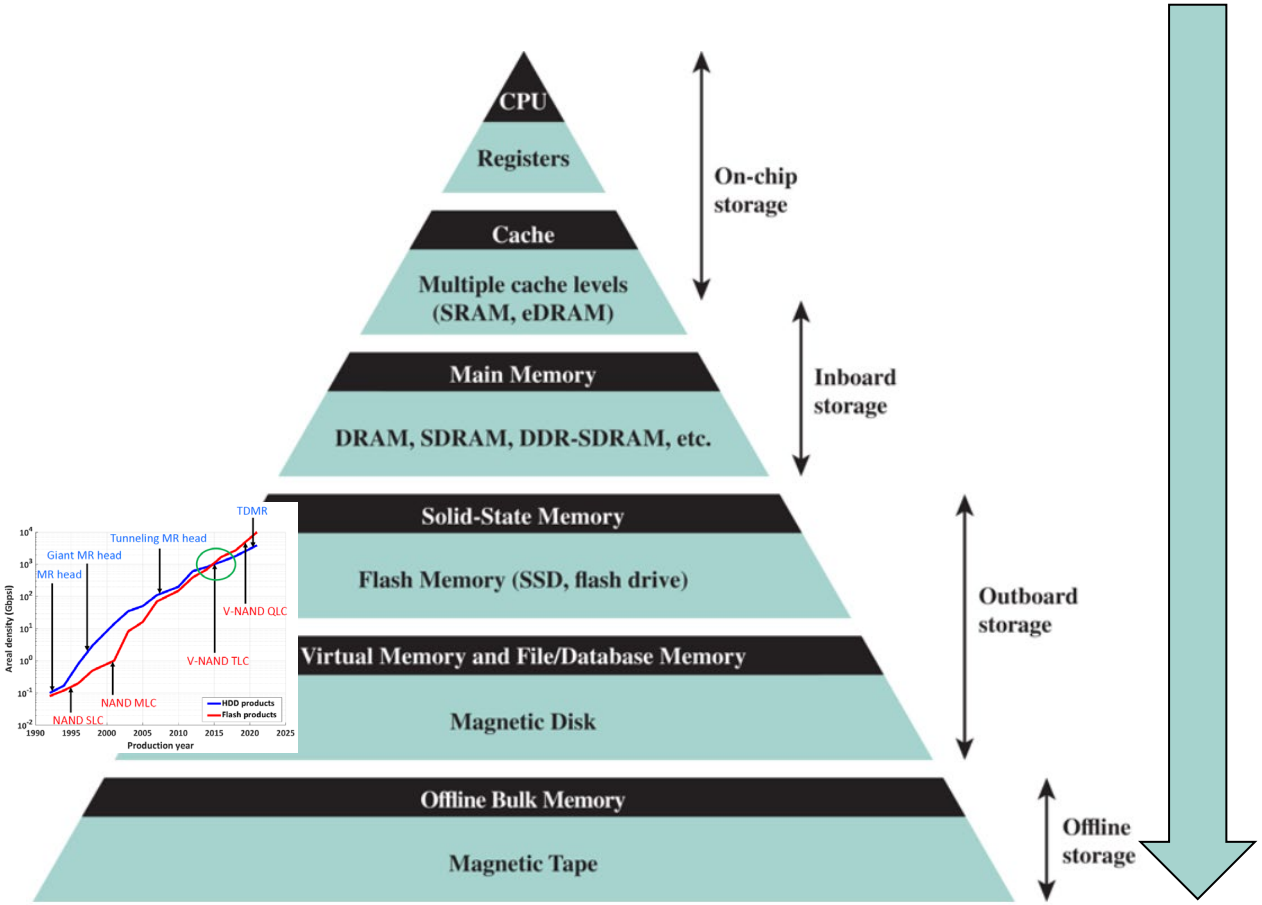
```
LDR Rd, [Rn, imm12]
STR Rd, [Rn, imm12]
ADD Rd, Rn, imm8
B BTA
```

Instruction Set



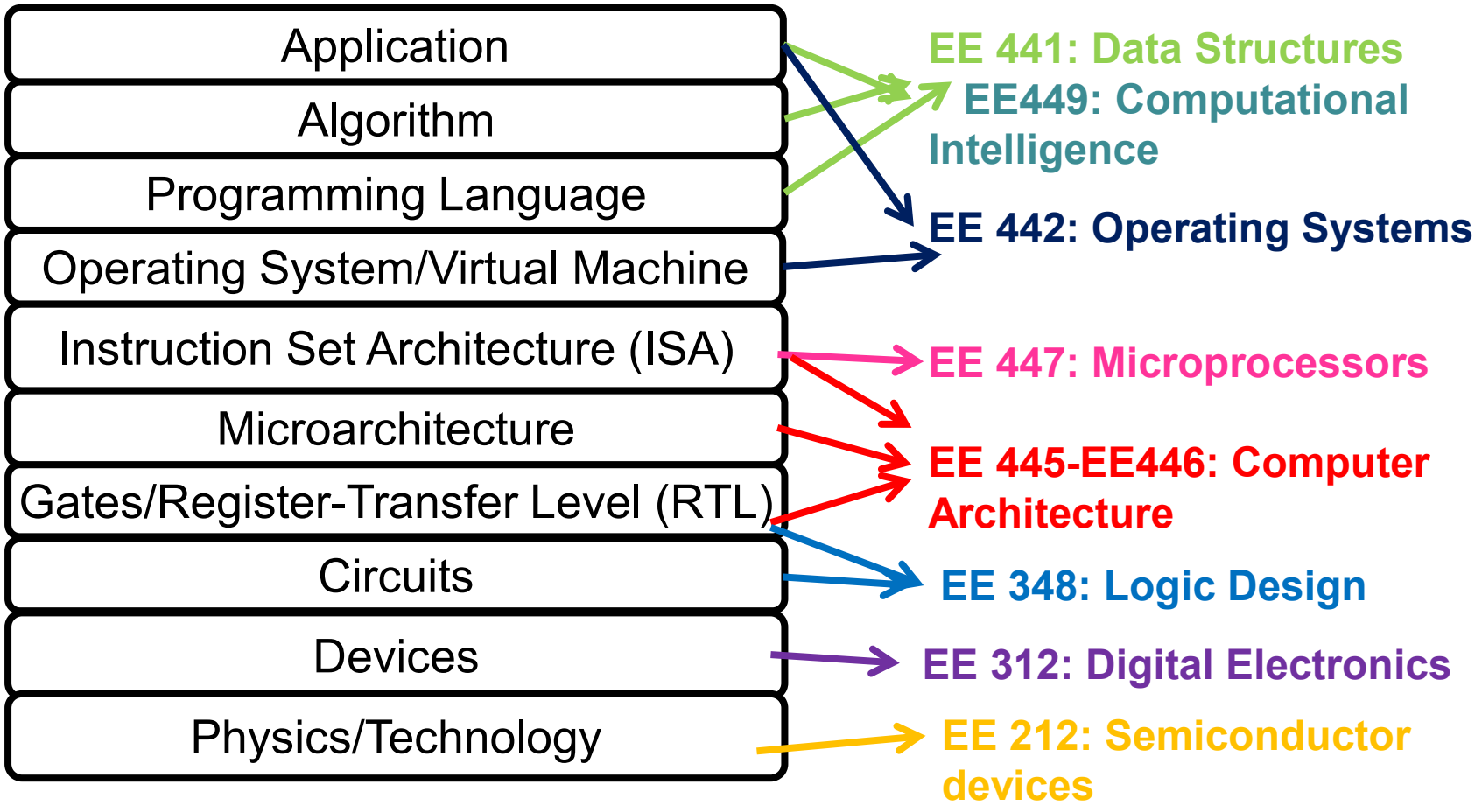
A Sneak Peek into EE446 ☺

Memory Hierarchy



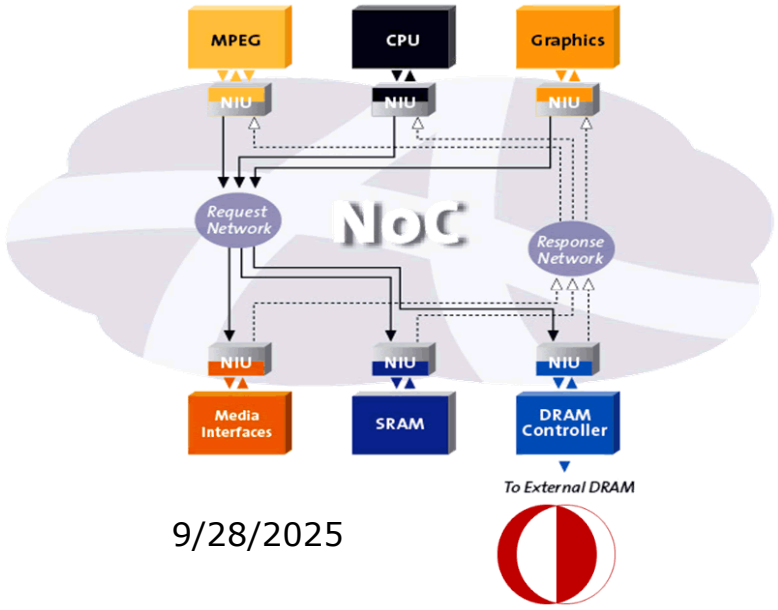
- Cost per byte decreases
- Average access time increases
- Average data transfer rate decreases
- Total memory size increases
- Frequency of access decreases ➔Principle of locality
- Data contained in a lower level are a superset of the next higher level ➔ Inclusion property

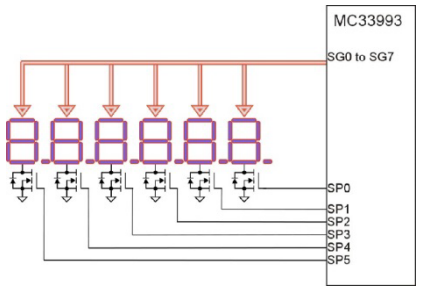
Computer Architecture in METU EE



Computers/Computer Architecture in METU EE

EE 444: Computer Networks





EE 447: Microprocessors: I/O device interfacing

Many computing Devices

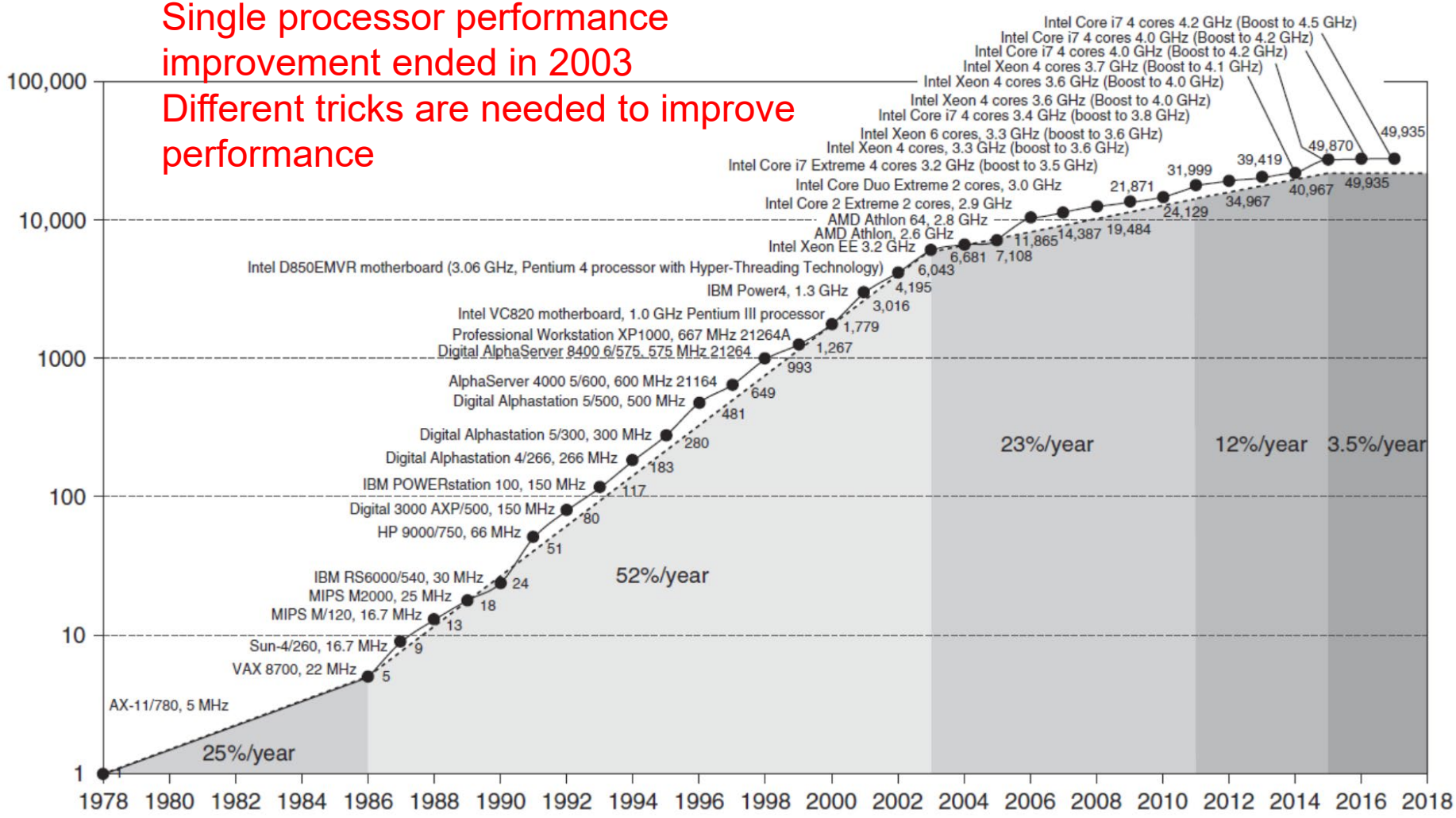
Classes of Computers

- Personal Mobile Device (PMD)
  - e.g. smart phones, tablet computers
  - Emphasis on energy efficiency and real-time
- Desktop Computing
  - Emphasis on price-performance
- Servers
  - Emphasis on availability, scalability, throughput
- Clusters / Warehouse Scale Computers
  - Used for “Software as a Service (SaaS)”
  - Emphasis on availability and price-performance
  - Sub-class: Supercomputers, emphasis: floating-point performance and fast internal networks
- Internet of Things/Embedded Computers
  - Emphasis: price

Trends and Performance

Single processor performance improvement ended in 2003

Different tricks are needed to improve performance



Trends and Performance

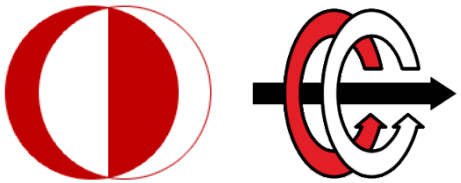
- Integrated circuit technology (Moore's Law)
  - Transistor density: 35%/year
  - Die size: 10-20%/year
  - Integration overall: 40-55%/year
- DRAM capacity: 25-40%/year (slowing)
  - 8 Gb (2014), 16 Gb (2019), possibly no 32 Gb
- Flash capacity: 50-60%/year
  - 8-10X cheaper/bit than DRAM
- Magnetic disk capacity: recently slowed to 20%/year
  - Density increases may no longer be possible (TDMR is an exception), maybe increase from 7 to 9 platters
  - 8-10X cheaper/bit then Flash
  - 200-300X cheaper/bit than DRAM

Performance

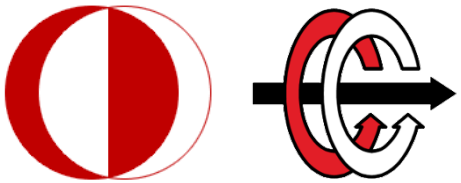
- Legacy view of computer architecture:
  - Instruction Set Architecture (ISA) design
  - i.e. decisions regarding:
    - registers, memory addressing, addressing modes, instruction operands, available operations, control flow instructions, instruction encoding
- Now computer architecture also focuses on:
  - Specific requirements of the target machine
  - Design to maximize performance within constraints: cost, power, and availability
  - ISA, microarchitecture, hardware

Performance Metrics

- Typical performance metrics:
  - Response time: Time between start and completion of an event
  - Throughput: Total work done in a given time
- Speedup of X relative to Y
  - Execution time$_Y$ / Execution time$_X$
- Execution time
  - Wall clock time: includes all system overheads
  - CPU time: only computation time
- Benchmarks
  - Kernels (e.g. matrix multiply)
  - Toy programs (e.g. sorting)
  - Synthetic benchmarks (e.g. Dhrystone)
  - Benchmark suites (e.g. SPEC06fp, TPC-C)



Introduction to Computer Architecture



EE348 Review

Representing Binary Numbers in Registers

- Definitions
  - Binary cell: a device that posses two stable states and is capable of storing one bit of information (0 or 1)
  - Register: a group of binary cells
  - Example: Register with n binary cells (bits)

X_{n-1}	X_{n-2}	. . .	X_2	X_1	X_0

- Facts
  - A register with n cells can be in one of 2^n possible states
  - If the register holds an unsigned integer, the possible values are from 0 to 2^n-1 (all bits are used for the magnitude of the integer)
  - If we want to represent signed integers, we need to use some bit to indicate if the integer is positive or negative

→ We will use the most significant bit (MSB) for this purpose

Signed Binary Numbers: Signed 2's Complement

- Definition for an n-bit register
  - If the MSB is 0, then the number is positive and the magnitude is directly given by the number
  - If the MSB is 1, then the number is negative and the magnitude is the 2's complement of the entire register content $(2^n - N)$
- Range
  - n bit register with value x: $-2^{n-1} \leq x \leq (2^{n-1} - 1)$
  - Example: n = 5: $10000 \leq x \leq 01111 \rightarrow -16 \leq x \leq +15$
- Examples
  - 0101 = +5 00111 = 7
  - 01001 = +9 11001 = -7

Number sign is changed by taking 2's complement of the entire number

n=5, 100000-00111=011001

00000 = 0: 1 zero representation

n=5, 100000-00000=100000

Arithmetic Addition: 2's complement

- Addition R = M + N
  - $-(2^{n-1}) \leq x \leq (2^{n-1} - 1)$ for x = R, M, N
- Examples (5-bit register)
  - M=+9 and N = +4

	0	1	0	0	1	+9
+	0	0	1	0	0	+4
	0	1	1	0	1	+13

  - M = -9 and N = +4

	1	0	1	1	1	-9
+	0	0	1	0	0	+4
	(1)	1	0	1	1	-5

Negative: → find magnitude from 2's complement: 0101 = 5

M = +9 and N = -4

	0	1	0	0	1	+9
+	1	1	1	0	0	-4
discard (1)	0	0	1	0	1	+5

M = -9 and N = -4

	1	0	1	1	1	-9
+	1	1	1	0	0	-4
discard (1)	(1)	0	0	1	1	-13

Negative: → find magnitude from 2's complement: 1101 = 13

Number of bits

- Observation
  - Overflow happens if the number of bits in a register is not sufficient

  → Number of bits should be increased if necessary
- Procedure for extension to a larger number of bits
  - Extend by adding bits to the left (MSBs)
  - Copy the value of the sign bit to the added MSB's

  → Number value does not change (positive and negative)
- Examples
  - 4-bit representation of 3 = 0011

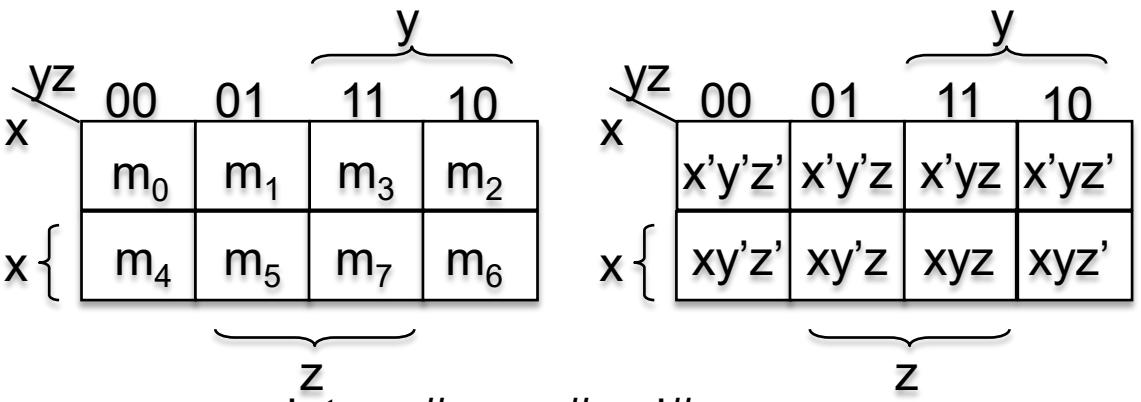
  → 8-bit sign-extended value: 00000011:
  - 4-bit representation of -5 = 1011

  → 10-bit sign-extended value: 1111111011

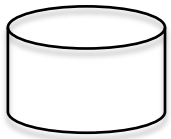
Gate Level Simplification: General K-Map

- K-Map for n-variable function
  - Cover all 1's in adjacent minterms with rectangles of 1, 2, 4,…,2^n squares
  - OR of two adjacent n-variable minterms leads to (n-1)-variable AND term
  - OR of 2^k adjacent n-variable minterms results in (n-k)-variable AND term

  → Always cover squares with the largest possible rectangles
- Representation of 3-variable K-Map (8 minterms)



minterm#= row# col#



The map rolls around
First and last columns are adjacent
m_0 m_2 m_4 m_6 are adjacent
x'y'z'+x'yz'+xy'z'+xyz'
=z'(x'y'+x'y+xy'+xy)
=z'

- Note: Gray code placement of columns

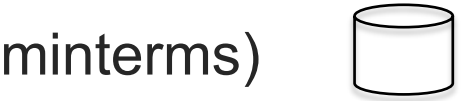
Simplification: Four-variable K-Map

- Representation of 4-variable K-Map (16 minterms)

minterm# =
row# col#

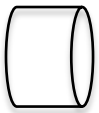
wx \ yz	00	01	11	10
00	w'x'y'z'	w'x'y'z	w'x'yz	w'x'yz'
01	w'x y'z'	w'xy'z	w'xyz	w'xyz'
11	wxy'z'	wxy'z	wxyz	wxyz'
10	wx' y'z'	wx'y'z	wx'yz	wx'yz'

y (columns 11, 10); x (rows 01, 11); w (rows 11, 10); z (columns 11, 10)



wx \ yz	00	01	11	10
00	m_0	m_1	m_3	m_2
01	m_4	m_5	m_7	m_6
11	m_{12}	m_{13}	m_{15}	m_{14}
10	m_8	m_9	m_{11}	m_{10}

y (columns 11, 10); x (rows 01, 11); w (rows 11, 10); z (columns 11, 10)



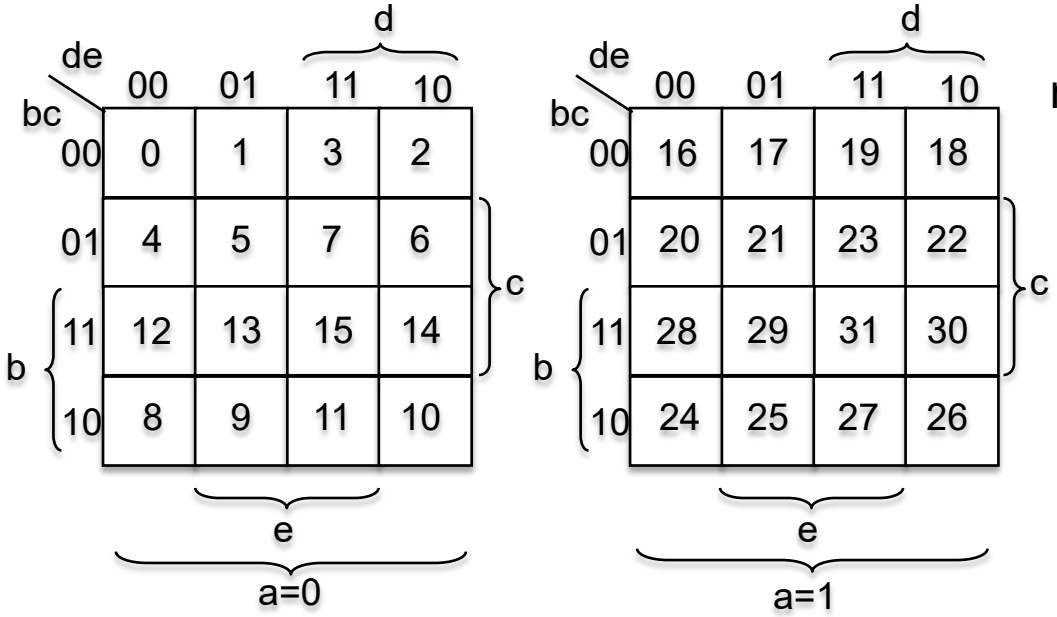
- Note the gray code placement of rows and columns
- 2^k adjacent minterms simplify to a term of n-k literals, here n=4

→ If k=4, then all squares have 1: F = 1

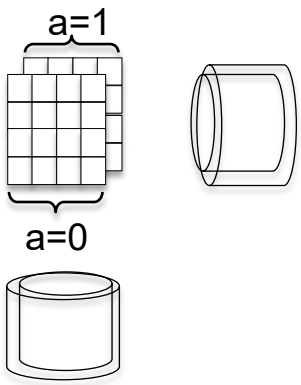
- the first and last columns/ first and last rows are adjacent
- m_0, m_2, m_8 and m_{10} are adjacent: w'x'y'z'+w'x'yz'+wx'y'z'+wx'yz'=x'z'

Simplification: 5-variable K-Map

- Representation of 5-variable K-Map (32 minterms)

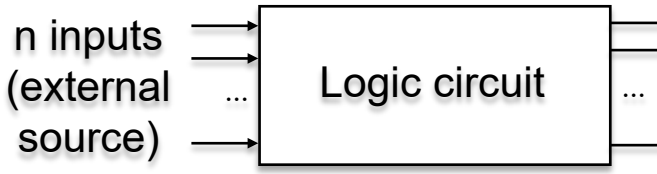


minterm#= layer# row# col#



- Cells in the same position in layers a=0 & a=1 are adjacent
  → m_k is adjacent to m_{k+16}: Example: m_3 is adjacent to m_{19}
- First and last rows/columns in layer a=0 and layer a = 1 are adjacent
- 2^k adjacent minterms lead to AND term of n-k literals (here n=5)

Logical Circuits: Combinational and Sequential



- Combinational circuits
  - Logic circuit outputs at the current time are determined from the current combination of the input
  - → Combinational circuits only use logic gates
- Sequential circuits
  - Logic circuit outputs at the current time are determined from the current and past combinations of the input
  - → Sequential circuits employ storage elements and logic gates
  - → Logic gate outputs are a function of the current inputs and the current state of the storage elements
  - → States of storage elements keep memory: they are a function of their previous states and previous inputs

Combinational Circuits: Full Adder

- Full Adder: combinational circuit that performs addition of 3 bits
  → 2 bits and carry
  - Inputs: a: augend, b: addend
    C_{in}: carry in
  - Outputs: s: sum, C_{out}: carry out
- Ouput from K-map
  - $S = a \oplus b \oplus C_{in}$
  - $C_{out} = ab + ab'C_{in} + a'bC_{in}$
    $= ab + (ab' + a'b)C_{in}$
    $= ab + (a \oplus b)\, C_{in}$
  - Advantage of this output
  → $a \oplus b$ is used in both outputs

$a+b+C_{in}$	sum	C_{out}
0+0+0=	0	0
0+0+1=	1	0
0+1+0=	1	0
0+1+1=	0	1
1+0+0=	1	0
1+0+1=	0	1
1+1+0=	0	1
1+1+1=	1	1

a	b	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

S:

a \ b	0	1	0	1
1	0	1	0	1

b; a; C_{in}

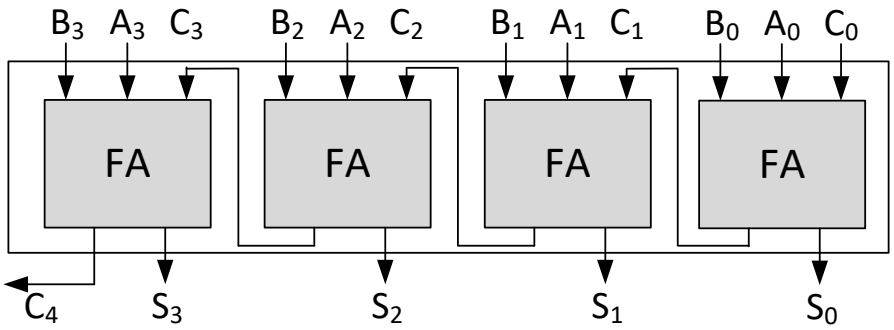
C_{out}:

a \ b	0	0	1	0
0	0	1	1	1

b; a; C_{in}

MSI LSI: 8 Bit Adder Design

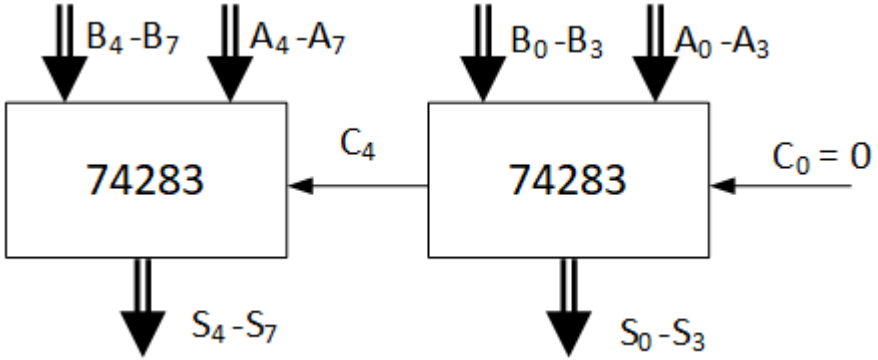
- 4 bit parallel adder from before



74 283
IC

Note: Direct design needs 2^9 input combinations!

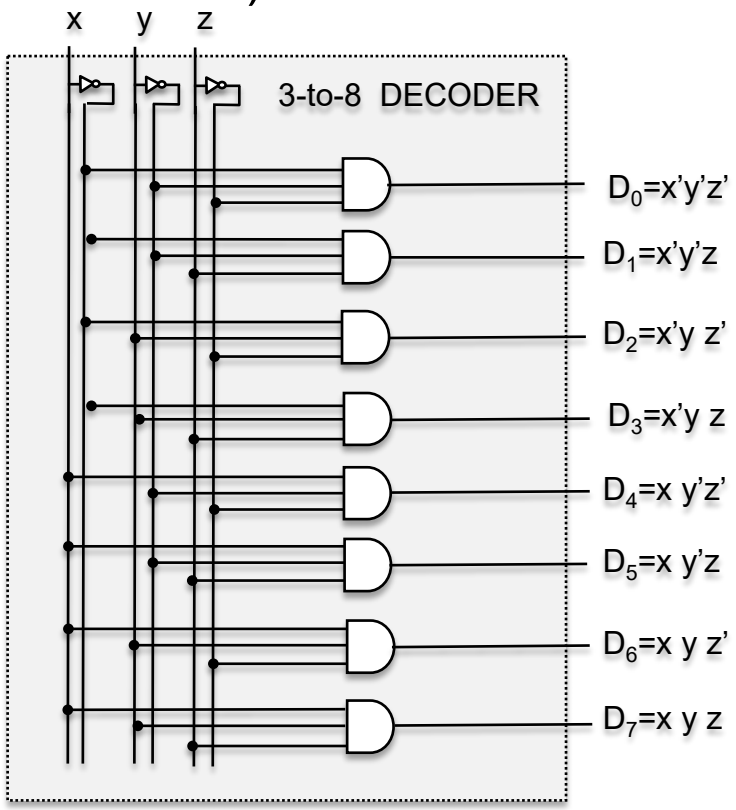
- 8 bit parallel adder: Use two 4 bit adders back to back



Decoder

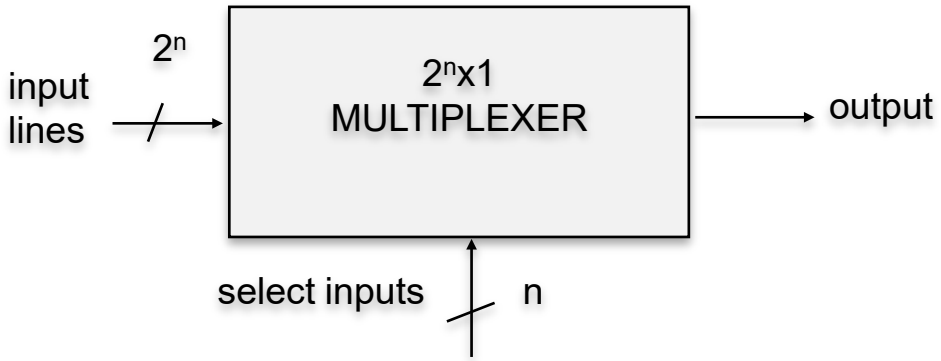
- (Line) Decoder: minterm generator
  - n inputs and 2^n outputs (one output per minterm)
  - Truth table for n = 3

x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

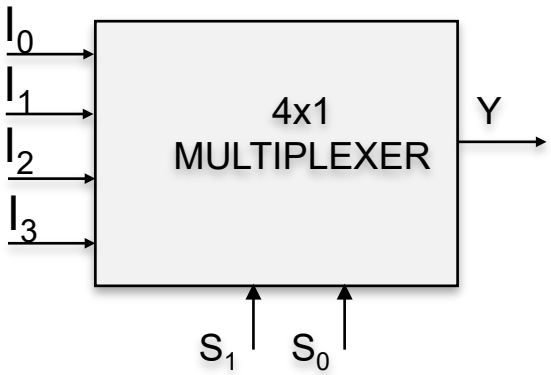


  - Each one of 8 AND gates generates one of the minterms
  - Only one output is 1 at a time

Multiplexer



- Multiplexer definition
  - Combinational circuit with 2^n inputs, n select inputs and 1 output
  - Receives information from 2^n input lines
  - Directs one input line to the single output line
  - Selection of the input line is controlled by the bit values of the n selection inputs
- 4x1 Multiplexer
  - 2^2 = 4 input lines, 2 select inputs
  - 1 output line
  - Example selections
    - $S_1S_0 = 01 \rightarrow Y = I_1$
    - $S_1S_0 = 11 \rightarrow Y = I_3$

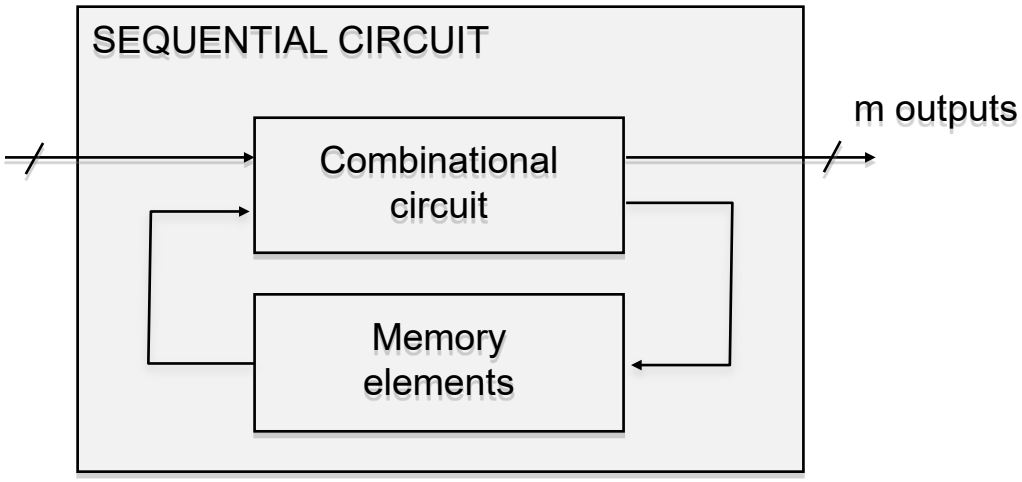


4 to 1 Multiplexer		
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

$$Y=S_1'S_0'I_0+ S_1'S_0I_1 + S_1S_0'I_2 + S_1S_0I_3$$

Sequential Circuits: Basics

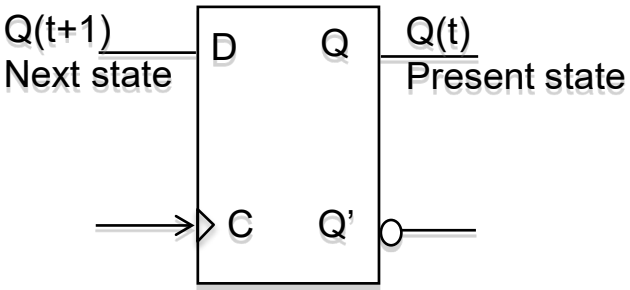
- Sequential circuit components
  - Combinational circuit
  - Memory elements which form feedback
- Memory elements
  - Devices capable of storing binary information within them
- Sequential circuit state
  - Binary information stored in the memory elements at any given time defines the **state** of the sequential circuit
  - State changes depending on the current state and the current input



Flip Flops: Input / Output Relation

- Definition
  - Present time: t
  - Present state: Value of the FF output at the present time t
    → we write Q(t)
  - The next state Q(t+1) is ready
    at the FF inputs at t
  - At t+1 (when the clock edge comes)
    the next state appears at Q

Example: D FF

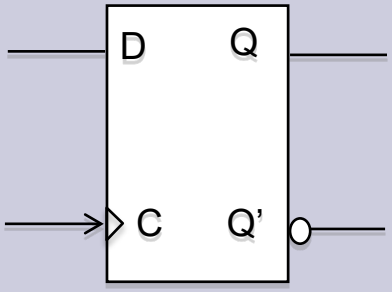


- Relation
  - Q(t+1) can be represented algebraically in terms of its present state Q(t) and present inputs (D(t) or R(t), S(t) → we write D, R, S)
    → Q(t+1) = f(Q(t),D) for D Flip Flop
    → Q(t+1) = f(Q(t),R,S) for RS Flip Flop

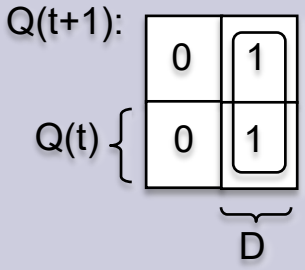
D Flip Flop: Formalization

Characteristic Equation, Characteristic and Excitation Tables

D FF



Simplification



Characteristic Table

D	Q(t)	Q(t+1)
0	0	0
0	1	0
1	0	1
1	1	1

Characteristic Table (simplified since Q(t+1) does not depend on Q(t))

D	Q(t+1)	Comment
0	0	clear (reset)
1	1	set

Characteristic Equation

Q(t+1) = D(t)

Excitation Table

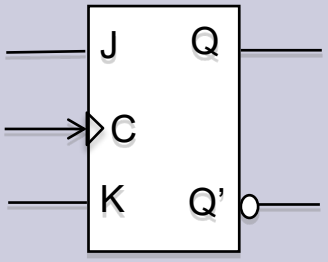
Q(t)	Q(t+1)	D
0	0	0
0	1	1
1	0	0
1	1	1

(describes required input D to get a certain output Q(t+1) from a given Q(t))

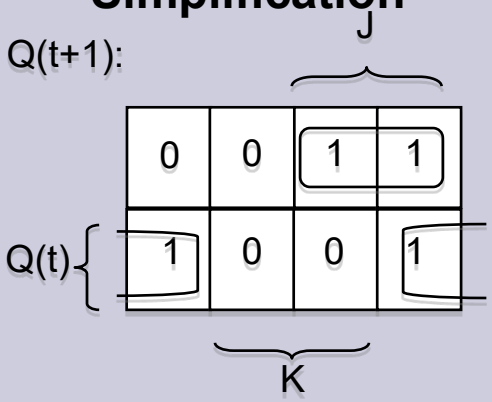
JK Flip Flop: Formalization

Characteristic Equation, Characteristic and Excitation Tables

JK FF



Simplification



Characteristic Table

J	K	Q(t)	Q(t+1)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Characteristic Equation

$$Q(t+1)=JQ'(t)+ K'Q(t)$$

Characteristic Table (simplified)

J	K	Q(t+1)	Comment
0	0	Q(t)	N.C
0	1	0	Reset
1	0	1	Set
1	1	Q'(t)	Toggle

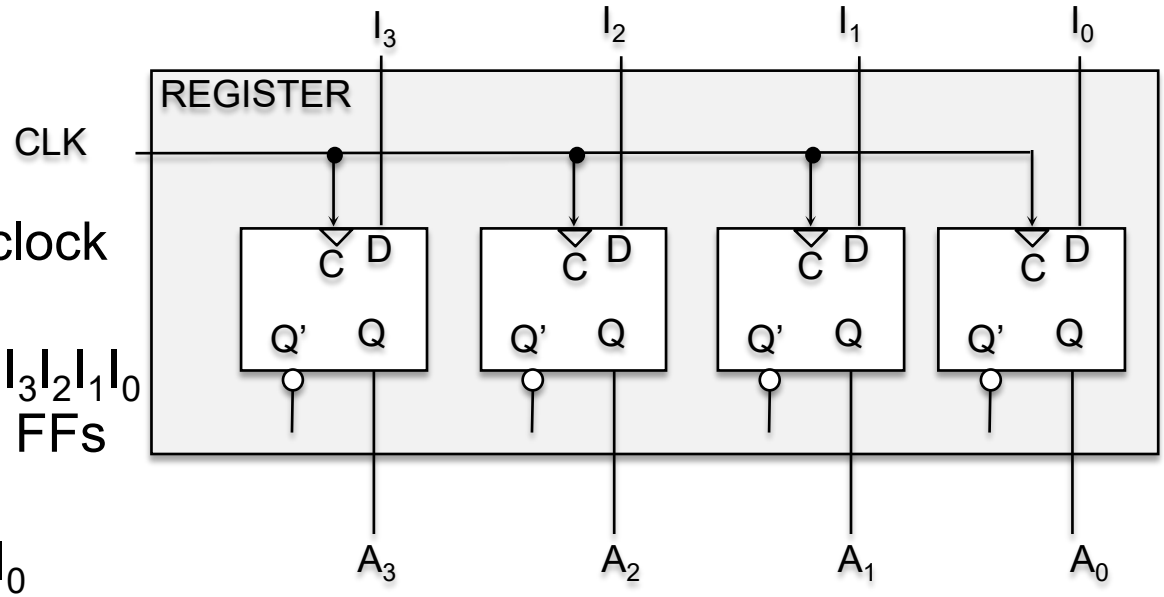
Excitation Table

Q(t)	Q(t+1)	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

Registers: Basics

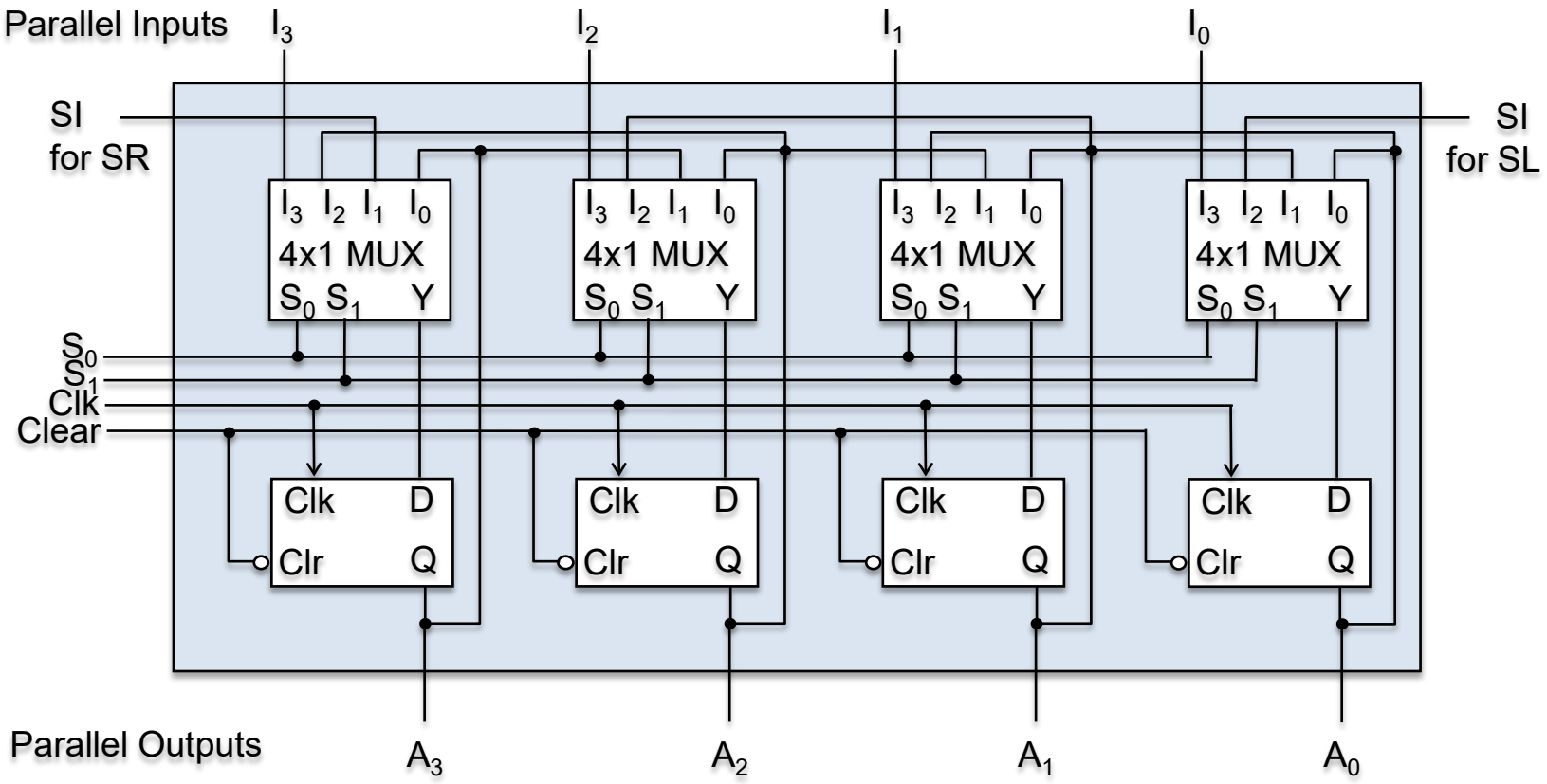
I_i	$A_i(t)$	$A_i(t+1)$
0	X	0
1	X	1

- n bit Register Components
  - A group of n FFs holding n bits of information
  - A combinational circuit determining how information is transfered into FFs
- Example: The simplest 4 bit Register
  - 4 bits of input information $I_3I_2I_1I_0$
  - 4 FFs
  - Output $A_3A_2A_1A_0$
  - When the next (positive/negative) clock edge comes
    → 4 bit-information $I_3I_2I_1I_0$ is loaded into the FFs of the register
    → $A_3A_2A_1A_0 = I_3I_2I_1I_0$



Registers: Bidirectional

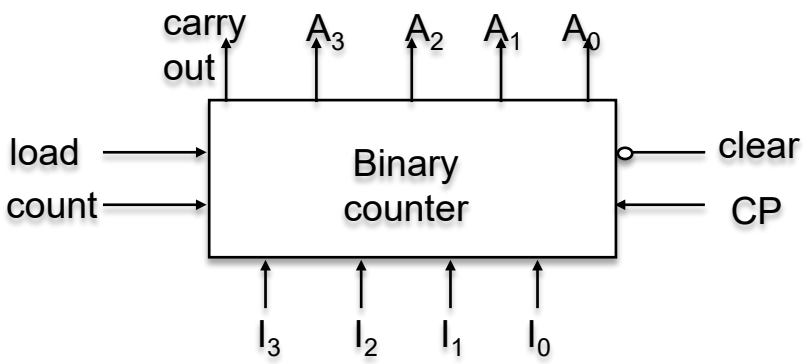
Operation	MUX Select S_1S_0	MUX out $A_i(t+1)$	
No change	00	A_i	
Shift Right (SR)	01	A_{i+1}	(Serial input for SR $\rightarrow A_3$)
Shift Left (SL)	10	A_{i-1}	(Serial input for SL $\rightarrow A_0$)
Parallel input	11	I_i	



Synchronous Counters: Parallel Load

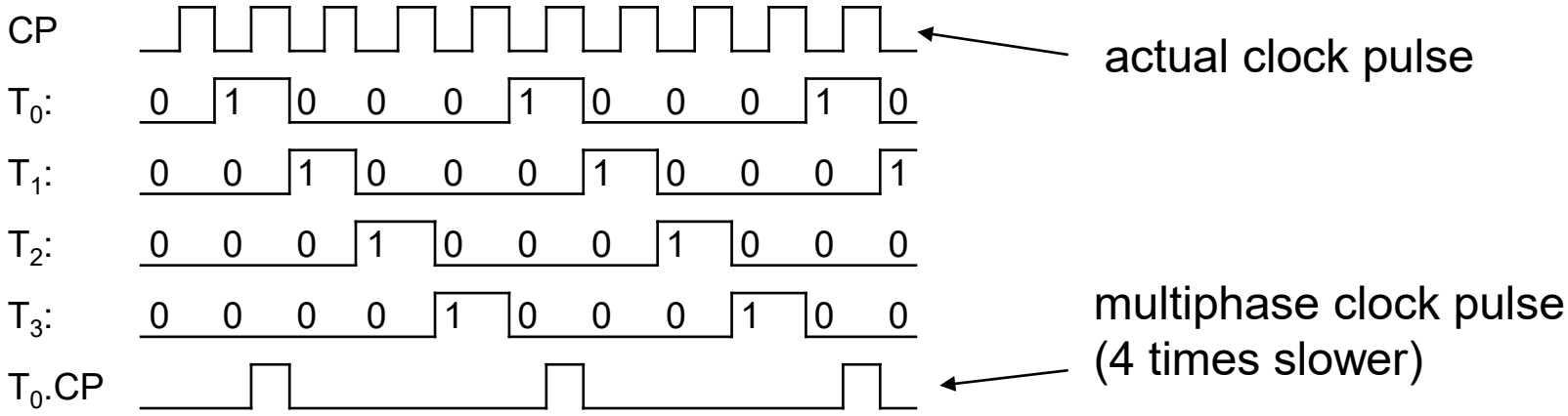
- Binary 4-bit counter with parallel load
  - Asynchronous reset (Clear = 0)
  - No change if Load = Count = 0
  - Parallel load at clock pulse if Load = 1 (independent of Count)
  - Count at clock pulse if Load = 0 and Count = 1
  - Carry out = 1 if counter value goes to 0
    → Carry out = $A_3A_2A_1A_0$

Clear	CP	Load	Count	Function
0	x	x	x	Clear to 0
1	x	0	0	No change
1	↑	1	x	Load Input
1	↑	0	1	Count next binary state

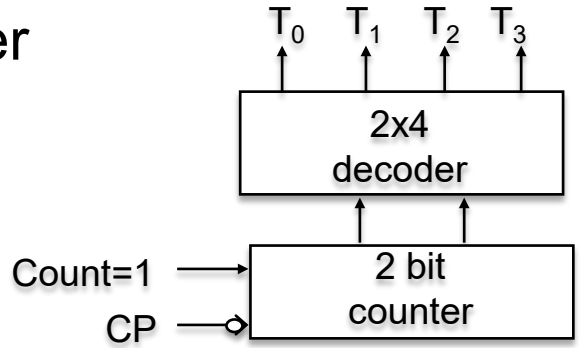


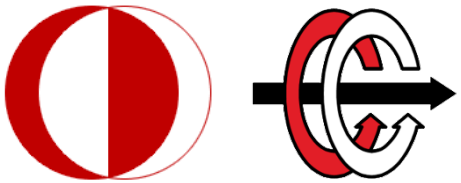
Timing Signals: Multiphase

- Multiphase clock pulses
  - Clock pulse is generated in multiples of the actual clock pulse



- Realization 2: 2 bit counter and 2x4 decoder
  - Decoder outputs are turned on sequentially based on the counter value





EE348 Review