

Final Exam Review

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Transistor: Digital Building Blocks

- Logically, each transistor acts as a switch
- Combined to implement logic functions (gates)
 - AND, OR, NOT
- Combined to build higher-level structures
 - Multiplexer, decoder, register, memory ...
 - Adder, multiplier ...
- Combined to build simple processor
 - LC-3

CS270 - Fall Semester 2016 2

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

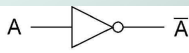
NOR Gate

A	B	C
0	0	1
0	1	0
1	0	0
1	1	0


Note: Serial structure on top, parallel on bottom. 3-3

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.


Basic Logic Gates




NOT



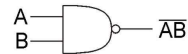
OR



NOR



AND



NAND

CS270 - Fall Semester 2016 4

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.


Propagation Delay

- Each gate has a propagation delay, typically fraction of a nanosecond (10^{-9} sec).
- Delays accumulate depending on the chain of gates the signals have to go through.
- Clock frequency of a processor is determined by the delay of the longest combinational path between storage elements, i.e. cycle time.


CS270 - Fall Semester 2016 5

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.


Boolean Algebra




$x \cdot 0 = 0$




$x \cdot 1 = x$




$x \cdot \bar{x} = 0$



$x + 0 = x$



$x + 1 = 1$



$x + \bar{x} = 1$

Remember Identify, Domination, Negation Laws from Logic!

CS270 - Fall Semester 2016

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

DeMorgan's Law

- Converting AND to OR (with some help from NOT)
- Consider the following gate:

*To convert AND to OR
(or vice versa),
invert inputs and output.*

A	B	\bar{A}	\bar{B}	$\bar{A} \cdot \bar{B}$	$\overline{A \cdot B}$
0	0	1	1	1	0
0	1	1	0	0	1
1	0	0	1	0	1
1	1	0	0	0	1

Same as A OR B!

CS270 - Fall Semester 2016 7

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Combinational Logic

- Cascading set of logic gates

What is the truth table?

CS270 - Fall Semester 2016 8

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Truth Table (from circuit)

- Truth table for circuit on previous slide

A	B	C	W	X	Y	Z
0	0	0	0	0	0	1
0	0	1	0	1	1	1
0	1	0	0	1	1	1
0	1	1	0	1	1	1
1	0	0	0	0	0	1
1	0	1	0	1	1	1
1	1	0	1	1	0	0
1	1	1	1	1	0	0

CS270 - Fall Semester 2016 9

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Decoder

- n inputs, 2^n outputs
 - exactly one output is 1 for each possible input pattern

2-bit decoder

A circuit diagram of a 2-bit decoder. It has two inputs, A and B. Each input is connected to all four AND gates. The outputs are: 1, if AB=00; 1, if AB=01; 1, if AB=10; 1, if AB=11.

CS270 - Fall Semester 2016 10

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Multiplexer (MUX)

- n -bit selector and 2^n inputs, one output
 - output equals one of the inputs, depending on selector

A circuit diagram of a 4-to-1 MUX. It has four data inputs A, B, C, D and two selector inputs S₁ and S₀. Each data input is connected to an AND gate. The selector inputs are connected to the AND gates such that the output is A if S=00, B if S=01, C if S=10, and D if S=11. All four AND gate outputs are connected to a single OR gate.

A, if S=00
B, if S=01
C, if S=10
D, if S=11

4-to-1 MUX

CS270 - Fall Semester 2016 11

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

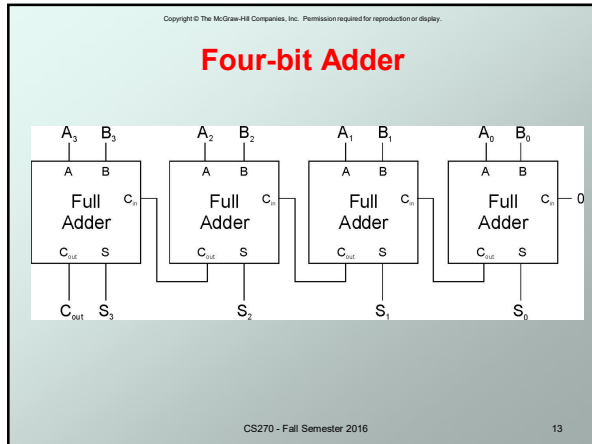
Full Adder

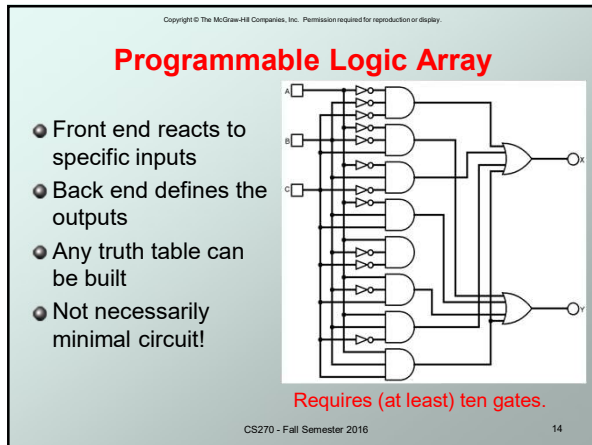
- Add two bits and carry-in, produce one-bit sum and carry-out.

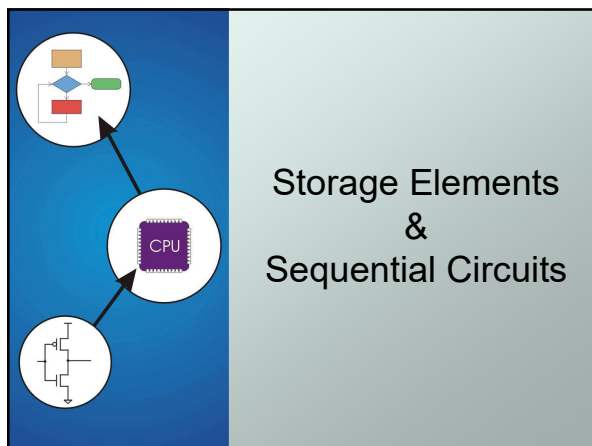
A circuit diagram of a full adder. It has three inputs: A, B, and C_{in}. It has two outputs: C_{out} and S. The circuit uses two half adders (each consisting of an XOR and an AND gate) and an OR gate. C_{out} is the OR of the two AND gates from the half adders. S is the XOR of the two XOR gates from the half adders.

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

CS270 - Fall Semester 2016 12







Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Combinational vs. Sequential

- **Combinational Circuit**
 - always gives the same output for a given set of inputs
 - ex: adder always generates sum and carry, regardless of previous inputs
- **Sequential Circuit**
 - stores information
 - output depends on stored information (state) plus input
 - so a given input might produce different outputs, depending on the stored information
 - *example*: ticket counter
 - advances when you push the button
 - output depends on previous state
 - useful for building “memory” elements and “state machines”

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

R-S Latch: Simple Storage Element

- R is used to “reset” or “clear” the element – set it to zero.
- S is used to “set” the element – set it to one.

S: 1, B: 0, A: 1, R: 1
a: 1, b: 0

S: 1, B: 1, A: 0, R: 1
a: 0, b: 1

- If both R and S are one, out could be either zero or one.
 - “quiescent” state -- holds its previous value
 - note: if a is 1, b is 0, and vice versa

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

R-S Latch Summary

- $R = S = 1$
 - hold current value in latch
- $S = 0, R = 1$
 - set value to 1
- $R = 0, S = 1$
 - set value to 0
- $R = S = 0$
 - both outputs equal one
 - final state determined by electrical properties of gates
 - *Don't do it!*

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Gated D-Latch

- Two inputs: D (data) and WE (write enable)
 - when WE = 1, latch is set to value of D
 - S = NOT(D), R = D
 - when WE = 0, latch holds previous value

3-19

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

2² x 3 Memory

3-21

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Finite State Machines

3-22

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

State Machine

- A general sequential circuit
 - Combines combinational logic with storage
 - “Remembers” state, and changes output (and state) based on **inputs and current state**

State Machine

Mealy type: general
 Moore type: Output depends only on state

3-23

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Finite State Machine

- A description of a system with the following components:
 1. A finite number of **states**
 2. A finite number of external **inputs**
 3. A finite number of external **outputs**
 4. An explicit specification of all **state transitions**
 5. An explicit specification of what determines each external **output value**
- Often described by a state diagram.
 - Inputs trigger state transitions.
 - Outputs are associated with each state (or with each transition).

3-24

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Mealy vs Moore state machines

- Moore: Outputs are only based on current state
 - Each state labeled with an output
 - Outputs change only at clock edge following input change
 - Potentially simpler to conceptualize
 - Simpler to interconnect with other state machines
 - Every Moore machine convertible to a Mealy machine
- Mealy: Outputs are based on current state and inputs
 - Each arc/transition labeled with a output
 - Tend to have fewer states
 - Outputs shown on transition arcs in state diagrams
 - Output changes in the same cycle as input is received

○ https://en.wikipedia.org/wiki/Mealy_machine
 ○ https://en.wikipedia.org/wiki/Moore_machine

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

State Diagram

- Shows **states** and **actions** that cause a **transition** between states.

3-26

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

The Clock

- Frequently, a **clock circuit** triggers transition from one state to the next.

- At the beginning of each clock cycle, state machine makes a transition, based on the current state and the external inputs.
 - Not always required. In lock example, the input itself triggers a transition.

3-27

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Storage: Master-Slave Flipflop

- A pair of gated D-latches, to isolate **next** state from **current** state.

During 1st phase (clock=1), previously-computed state becomes **current** state and is sent to the logic circuit.

During 2nd phase (clock=0), **next** state, computed by logic circuit, is stored in Latch A.

CS270 - Fall Semester 2016 28

RTN / Von Neumann

CS270 - Fall Semester 2016 29

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

How does the LC-3 fetch an instruction?

Transfer the PC into MAR
 Cycle 1: $MAR \leftarrow PC$ # LD.MAR, GatePC

Read memory; increment PC
 Cycle 2: $MDR \leftarrow Mem[MAR]; PC \leftarrow PC+1$ # LD.MDR, MDR.SEL, MEM.EN, LD.PC, PCMUX

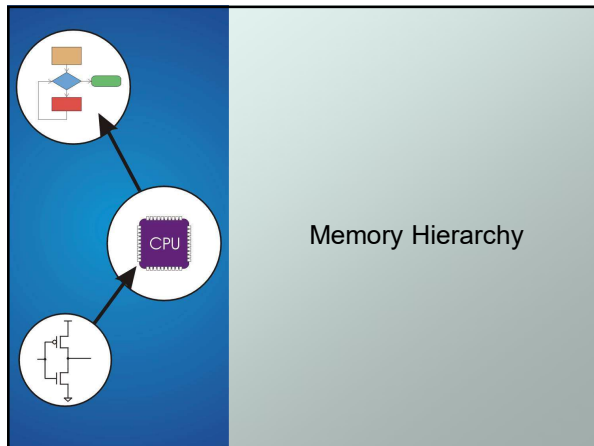
Transfer MDR into IR
 Cycle 3: $IR \leftarrow MDR$ # LD.IR, GATEMDR

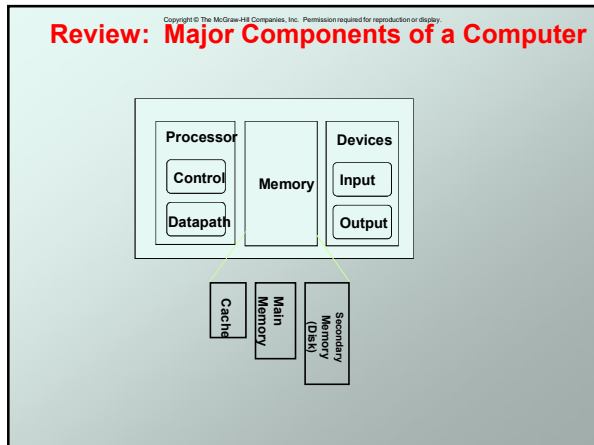
30

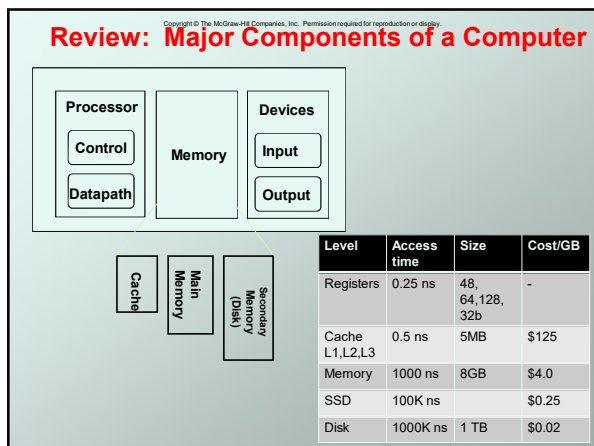
Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Von Neumann Model

4-31







Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

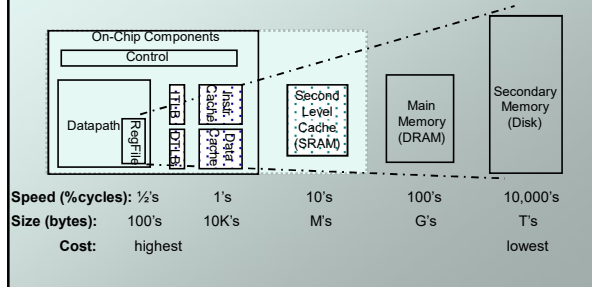
The Memory Hierarchy: Key facts and ideas (1)

- Programs keep getting bigger exponentially.
- Memory cost /bit
 - Faster technologies are expensive, slower are cheaper. Different by orders of magnitude
 - With time storage density goes up driving per bit cost down.
- Locality in program execution
 - Code/data used recently will likely be needed soon.
 - Code/data that is near the one recently used, will likely be needed soon.

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

A Typical Memory Hierarchy

- Take advantage of the **principle of locality** to present the user with as much memory as is available in the **cheapest** technology at the speed offered by the **fastest** technology



Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Principle of Locality

- Programs access a small proportion of their address space at any time
- Temporal locality
 - Items accessed recently are likely to be accessed again soon
 - e.g., instructions in a loop, induction variables
- Spatial locality
 - Items near those accessed recently are likely to be accessed soon
 - E.g., sequential instruction access, array data

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Cache Misses

- On cache hit, CPU proceeds normally
- On cache miss
 - Stall the CPU pipeline
 - Fetch block from next level of hierarchy
 - Instruction cache miss
 - Restart instruction fetch
 - Data cache miss
 - Complete data access

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Multilevel Caches

- Primary cache attached to CPU
 - Small, but fast
- Level-2 cache services misses from primary cache
 - Larger, slower, but still faster than main memory
- Main memory services L-2 cache misses
- Some systems now include L-3 cache

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

The Memory Hierarchy

The BIG Picture

- Common principles apply at all levels of the memory hierarchy
 - Based on notions of caching
- At each level in the hierarchy
 - Block placement
 - Finding a block
 - Replacement on a miss
 - Write policy

Chapter 5 — Large and Fast: Exploiting Memory Hierarchy — 40

§ 5.8 A Common Framework for Memory Hierarchies

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Block Placement

- Determined by associativity
 - Direct mapped (1-way associative)
 - One choice for placement
 - n-way set associative
 - n choices within a set
 - Fully associative
 - Any location
- Higher associativity reduces miss rate
 - Increases complexity, cost, and access time

Chapter 5 — Large and Fast: Exploiting Memory Hierarchy — 41

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Finding a Block

Associativity	Location method	Tag comparisons
Direct mapped	Index	1
n-way set associative	Set index, then search entries within the set	n
Fully associative	Search all entries	#entries

- Hardware caches
 - Reduce comparisons to reduce cost

Chapter 5 — Large and Fast: Exploiting Memory Hierarchy — 42

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Replacement

- Choice of entry to replace on a miss
 - Least recently used (LRU)
 - Complex and costly hardware for high associativity
 - Random
 - Close to LRU, easier to implement

Chapter 5 — Large and Fast: Exploiting Memory Hierarchy — 43

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Write Policy

- Write-through
 - Update both upper and lower levels
 - Simplifies replacement, but may require write buffer
- Write-back
 - Update upper level only
 - Update lower level when block is replaced
 - Need to keep more state

Chapter 5 — Large and Fast: Exploiting Memory Hierarchy — 44

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Sources of Misses

- Compulsory misses (aka cold start misses)
 - First access to a block
- Capacity misses
 - Due to finite cache size
 - A replaced block is later accessed again
- Conflict misses (aka collision misses)
 - In a non-fully associative cache
 - Due to competition for entries in a set
 - Would not occur in a fully associative cache of the same total size

Chapter 5 — Large and Fast: Exploiting Memory Hierarchy — 45

Copyright © The McGraw-Hill Companies, Inc. Permission required for reproduction or display.

Cache Design Trade-offs

Design change	Effect on miss rate	Negative performance effect
Increase cache size	Decrease capacity misses	May increase access time
Increase associativity	Decrease conflict misses	May increase access time
Increase block size	Decrease compulsory misses	Increases miss penalty. For very large block size, may increase miss rate due to pollution.

Chapter 5 — Large and Fast: Exploiting Memory Hierarchy — 46

3-Level Cache Organization

	Intel Nehalem	AMD Opteron X4
L1 caches (per core)	L1 I-cache: 32KB, 64-byte blocks, 4-way, approx LRU replacement, hit time n/a L1 D-cache: 32KB, 64-byte blocks, 8-way, approx LRU replacement, write-back/allocate, hit time n/a	L1 I-cache: 32KB, 64-byte blocks, 2-way, LRU replacement, hit time 3 cycles L1 D-cache: 32KB, 64-byte blocks, 2-way, LRU replacement, write-back/allocate, hit time 9 cycles
L2 unified cache (per core)	256KB, 64-byte blocks, 8-way, approx LRU replacement, write-back/allocate, hit time n/a	512KB, 64-byte blocks, 16-way, approx LRU replacement, write-back/allocate, hit time n/a
L3 unified cache (shared)	8MB, 64-byte blocks, 16-way, replacement n/a, write-back/allocate, hit time n/a	2MB, 64-byte blocks, 32-way, replace block shared by fewest cores, write-back/allocate, hit time 32 cycles

n/a: data not available
