

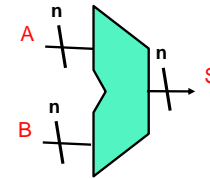
Basic Components

Based on slides © McGraw-Hill
Additional material © 2004/2005 Lewis/Martin
Modified by Diana Palsetia

Combinational: Adder

Adder

- A and B are operands
- S is the result of the addition
- Write now we are not interested in how it works (CIT 595 goes in detail)



Black box view of the Adder

CIT 593

Components of a Computer

Combinational Structures/Circuits

- Always gives the same output for a given set of inputs
- Do not store any information
- Examples: adder, subtractor and multiplexer (mux)

Sequential Structures/Circuits

- Always store information
- Example: memory, register file

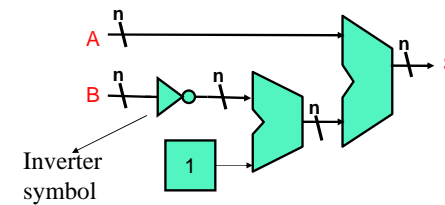
CIT 593

3-2

Combinational: Subtractor

Build a subtractor from an adder

- Calculate $A - B = A + -B$
- Recall 2's complement!!
- Negate B
- Recall $-B = \text{NOT}(B) + 1$



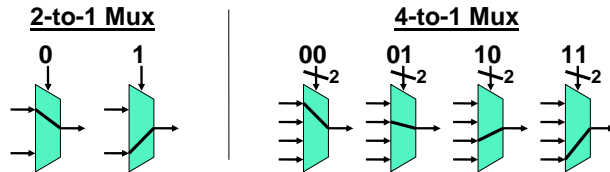
CIT 593

3-4

Combinational: Multiplexer (MUX)

Selector/Chooser of signals

- Multi-way switch



In general

- N select bits chooses from 2^N inputs
- An incredibly useful building block

CIT 593

3-5

Sequential Structure/Circuit

Sequential Structure/Circuit

- Stores information
- **Output** depends on stored information (**state**) plus **input**
 - Given input might produce different outputs, depending on stored information
- **State** = snapshot of all relevant elements of system at moment snapshot is taken
- **Example:**
 - **state** = Score board of basketball game (number of points, time remaining, possession)
 - **input** = which team scored the point
 - **output** = point increase for the team that just scored

Useful for building “memory” elements and “state machines”

CIT 593

3-7

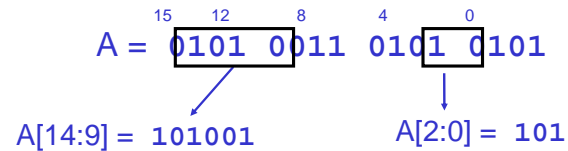
Aside: More on Representing Multi-bit Values

Number bits from right (0) to left (n-1)

- Just a convention -- could be left to right, but must be consistent

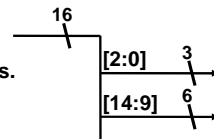
Use brackets to denote range:

$D[l:r]$ denotes bit l to bit r, from *left to right*



May also see $A<14:9>$

- Especially in hardware block diagrams.



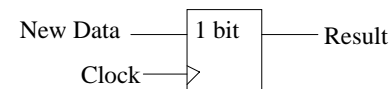
CIT 593

3-6

Fundamental Storage Unit

Storage Unit

- Stores 1 bit
- Data is changed on an event (a.k.a clock)
 - To have control: do not want to change data abruptly



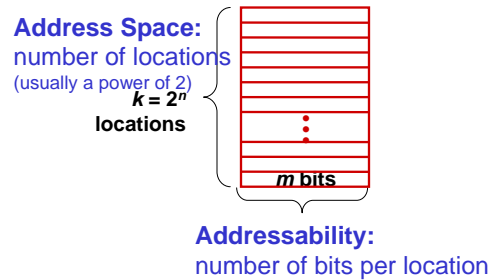
clock is a pulse that generated using electronic circuitry
if clock = '1' then new data stored
else if clock = '0' then the stored value remains.

CIT 593

Sequential: Memory

Memory (logical view)

- Holds information (e.g. phone directory contains phone numbers)
- a logical k by m array of stored bits

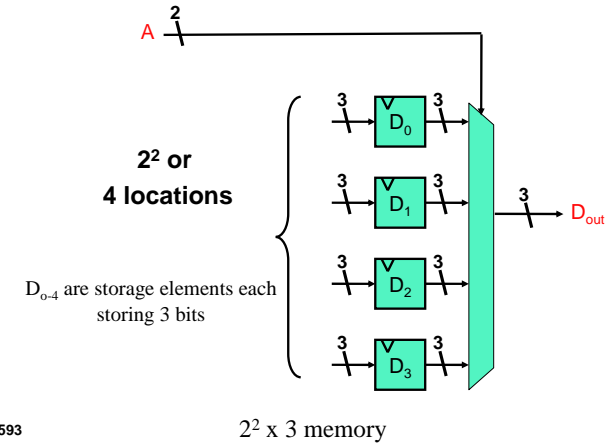


CIT 593

3-9

Sequential: Memory (1 layer down)

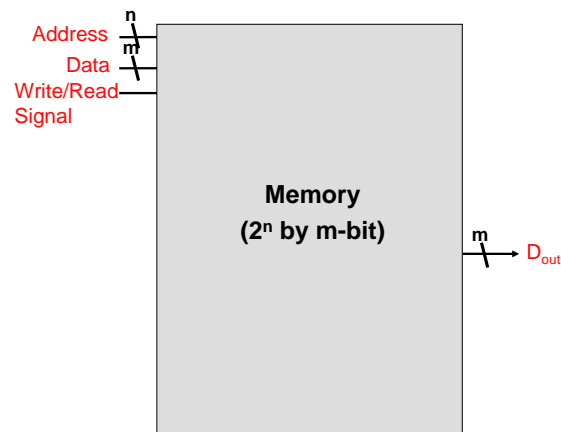
Comprises of storage units and a mux



CIT 593

3-11

Sequential: Memory as a black box



CIT 593

3-10

Memory Size => how many locations? How much information?

Terminology First

- Decimal system short form for large denominations
 - > 1000 = 1K(thousand), 1,000,000 = 1M (million)
- Binary Systems also notation
 - > 1 Byte = $2^3 = 8$ bits
 - > 1 Kilo Byte (KB) = 1024 bytes = 2^{10} Bytes = $2^{10} \times 2^3 = 2^{13} = 8192$ bits
 - > 1 Mega Byte (MB) = 1024 KB = 2^{20} Bytes = $2^{20} \times 2^3 = 2^{23} = 8388608$ bits
 - > 1 Giga Byte (GB) = 1024 MB = 2^{30} Bytes = $2^{30} \times 2^3 = 2^{33} = 8589934592$ bits

Example: 16 MB memory = $2^4 \times 2^{20} = 2^{24}$ bytes

- $2^{24} = 16777216$ (~ 16 million) unique locations, each holds a 8-bit value

Different size of data storage

- We can also have each memory location hold 16-bit or 32-bit depending on the machine's need
- Example: LC3 has $2^{16} \times 16 (= 2^1)$ memory = $2^{17} \times 2^3 = 2^{20}$ bytes

CIT 593

Sequential: Register File

Small fast memory

- Small – only 8-32 locations. E.g. LC3 has only 8 registers
- Fast – For faster access compared to 256 MB memory
 - This all driven by technology and cost

Used for temporary storage

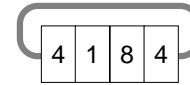
- To avoid going back and forth to the memory
- For intermediate calculations use smaller size
 - E.g. $(A+B)*C$

Analogy: we use our desk to keep a few books (immediate need), and use a bookshelf to store the other books.

CIT 593

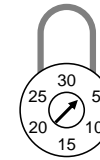
Combinational vs. Sequential Example

Two types of “combination” locks



Combinational

Success depends only on the **values**, not the order in which they are set.



Sequential

Success depends on the **sequence** of values (e.g. R-13, L-22, R-3).

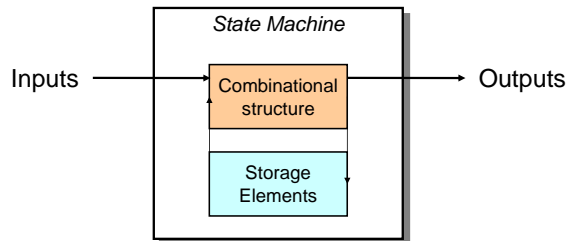
CIT 593

3-15

Sequential: State Machine

Another type of sequential structure

- Combines combinational structure with storage (memory element)
- “Remembers” state (old stored information), and changes output (and state) based on **inputs** and **current state**



CIT 593

3-14

State of Sequential Lock

Our lock example has four different states, labeled A-D:

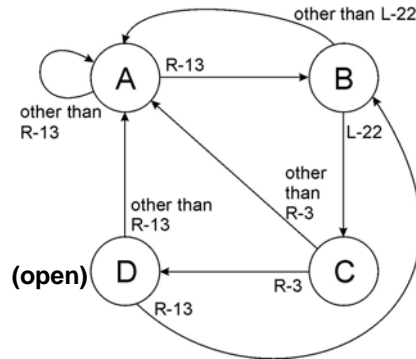
- A:** The lock is **not open**, and no relevant operations have been performed
- B:** The lock is **not open**, and the user has completed the **R-13** operation
- C:** The lock is **not open**, and the user has completed **R-13**, followed by **L-22**
- D:** The lock is **open**

CIT 593

3-16

Sequential Lock State Diagram

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



CIT 593

3-17

Next

Put all the components together and we have a working computer – chp 4

CIT 593