Chapter 3
Components

Arthur B. Maccabe
Department of Computer Science
The University of New Mexico

Slide 1

Overview

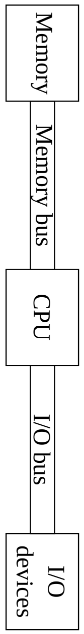
- component overview
- memory
  - operations
  - big and little endian
  - the memory hierarchy—caching
- the CPU
  - an accumulator machine
  - instruction set
  - state machine implementation
- I/O devices

  control, data, and status registers
- buses

  internal and external

Slide 2

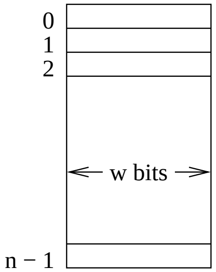
Basic Components



Slide 3

Basic Memory Characteristics

- memory is a collection of independent storage units, called "cells"
- *width* is the number of bits in each cell
- each cell has a unique *address* (e.g., $[0, n)$)
- an $n \times w$ memory



- *byte addressable*: each storage unit holds one byte (8 bits)
- *bit addressable*: each storage unit holds a single bit

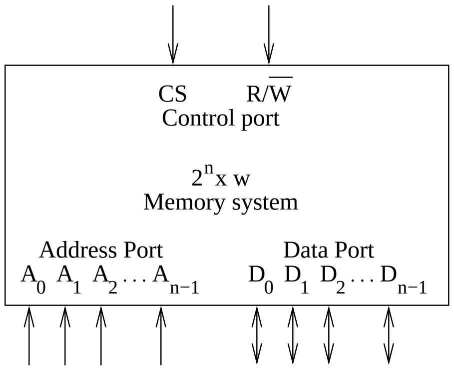
Slide 4

Memory Operations

- basic operations
  - write: store a value into a cell
  - read: return the most recently stored value
  - aka, store and fetch
- limited write operations
  - ROM (read-only memory)
  - PROM (programmable read-only memory)
  - EPROM (erasable PROM)
  - EEPROM (electrically erasable PROM)
- access patterns
  - sequential access
  - random access (RAM)
  - direct access
- volatility

Slide 5

Memory Ports



Notes	
CS	Chip select
R/$\overline{W}$	Read/not write
A	Address lines
D	Data lines

Slide 6

Steps in the Memory operations

- steps in a read operation
  1. Put the address value on $A_0 \ldots A_{n-1}$ and raise $R/\overline{W}$.
  2. Raise CS.
  3. Wait.
  4. Read the value on $D_0 \ldots D_{w-1}$.
  5. Drop CS and $R/\overline{W}$.
- steps in a write operation
  1. Put the address value on $A_0 \ldots A_{n-1}$ and the data value on $D_0 \ldots D_{w-1}$.
  2. Raise CS.
  3. Wait.
  4. Drop CS.
- *access time*: the time from the start of a read until the data is available
- *cycle time*: the time between consecutive memory operations

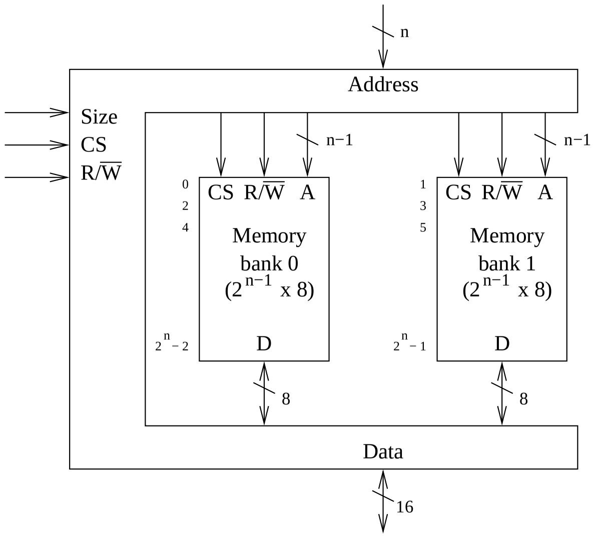
Slide 7

Memory Units (Bytes and Words)

- *logical word size*: memory width
- 8-bit characters and 16 (or more) bit numbers
  - 8-bit words: numeric operations require multiple reads/writes
  - 16-bit words: waste space for characters or pack characters
- modern memory systems provide access to 8-bit, 16-bit, 32-bit, and 64-bit values

Slide 8

Memory Banks



Slide 9

Terminology

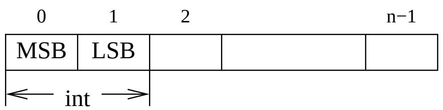
- *global* addresses are defined with respect to the entire memory system
- *relative* addresses are defined with respect to a memory bank.
- *interleaved addresses* alternate between the memory banks
- a *justified* data port always places byte values on the same data lines (independent of the address)
- in an *aligned* memory system, 16-bit values must have even addresses

Slide 10

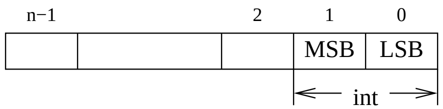
Byte Ordering

- labeling boxes, left-to-right or right-to-left?

A. Left−to−right labels (Big Endian)



B. Right−to−left labels (Little Endian)



- consistency: bit numbering is the same as byte numbering

Slide 11

The Memory Hierarchy

- example:
  - characteristics
    - ∗ 50MHz processor
    - ∗ 70nsec memory access
  - calculate processor cycles / memory access

    50MHz means 20nsec / processor cycle

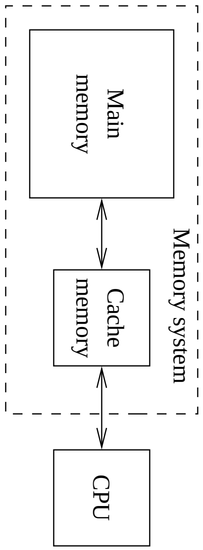
    $70/20 = 3.5$ processor cycles / memory access
- example:
  - characteristics
    - ∗ 200MHz processor
    - ∗ 50nsec memory access
  - calculate processor cycles / memory access

    200MHz means 5nsec / processor cycle

    $50/10 = 10$ processor cycles / memory access

Slide 12

A Single Level Cache



Slide 13

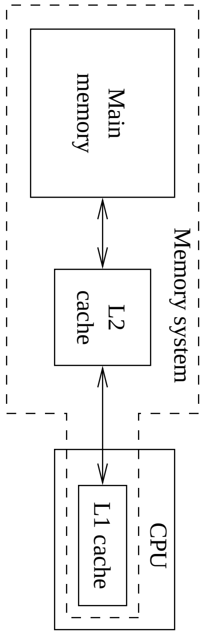
Cache Hit Rate and Effective Memory Access Time

- *cache hit rate*: the percentage of time that memory accesses are satisfied by a cache
- *effective memory access time*: the time that it takes to access a memory value given the cache hit rate
- example:
  - characteristics
    - 50nsec memory access time
    - 10nsec cache access time
    - 80% cache hit rate
  - calculate the effective memory access time

    80% of the accesses take 10nsec while 20% require 50nsec

    $$.80 \times 10 + .20 \times 50 = 8 + 10 = 18\text{nsec}$$

A Two Level Cache



Examples

- with an L2 cache
  - characteristics
    - 50nsec memory access time
    - 10nsec L2 cache access time
    - 5nsec L1 cache access time
    - 80% cache hit rate in both caches
  - calculate the effective memory access time

    80% of the accesses take 5nsec
    80% of the other 20% (16%) take 10nsec
    the remaining 4% take 50nsec

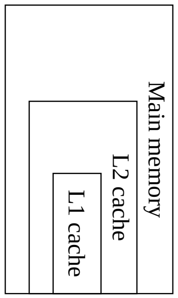
$$.80 \times 5 + .16 \times 10 + .04 \times 50$$
$$= 4 + 1.6 + 2 = 7.6\text{nsec}$$

- same calculation without an L2 cache

$$.80 \times 5 + .20 \times 50 = 4 + 10 = 14\text{nsec}$$

Slide 16

The Memory Hierarchy



Slide 17

The CPU

- data manipulation
- running example: an accumulator machine
- instruction set (architecture)
- implementation (organization)
  - registers and data paths
  - instruction interpretation

Slide 18

An Example Calculation

- the caluclation

$$47 * 82 + 91$$

- using an accumulator
  1. LOAD 47 into the accumulator.
  2. MULTIPLY the accumulator by 82.
  3. ADD 91 to the accumulator.
  4. EXAMINE the value in the accumulator.
- making the accumulator implicit
  1. LOAD 47
  2. MULTIPLY 82
  3. ADD 91
  4. EXAMINE

Slide 19

An Instruction Set

Symbolic	Semantics
ADD *addr*	Add the value in memory cell *addr* to the value in the accum.
SUB *addr*	Subtract the value in memory cell *addr* from the value in the accum.
MPY *addr*	Multiply the value in memory cell *addr* by the value in the accum. Load the accum with the least significant 8 bits of the result.
DIV *addr*	Divide the value in the accum by the value in memory cell *addr*. Load the accum with the integer portion of the result.
LOAD *addr*	Load the accum with the value in memory cell *addr*.
STORE *addr*	Store the value of the accum into memory cell *addr*.

Slide 20

An Example

- the calculation

$$c = a * b + c * d$$

- a, b, c, and d are stored in locations 20, 21, 22, and 23
- the code

LOAD	20;	$acc = a$
MPY	21;	$acc = a * b$
STORE	30;	$temp = a * b$
LOAD	22;	$acc = c$
MPY	23;	$acc = c * d$
ADD	30;	$acc = a * b + c * d$
STORE	22;	$c = a * b + c * d$

Slide 21

Instruction Encoding

- format

7 5	4 0
OPCODE	ADDRESS

- opcode encodings

Operation	Encoding
ADD	000
SUB	001
MPY	010
DIV	011
LOAD	100
STORE	101

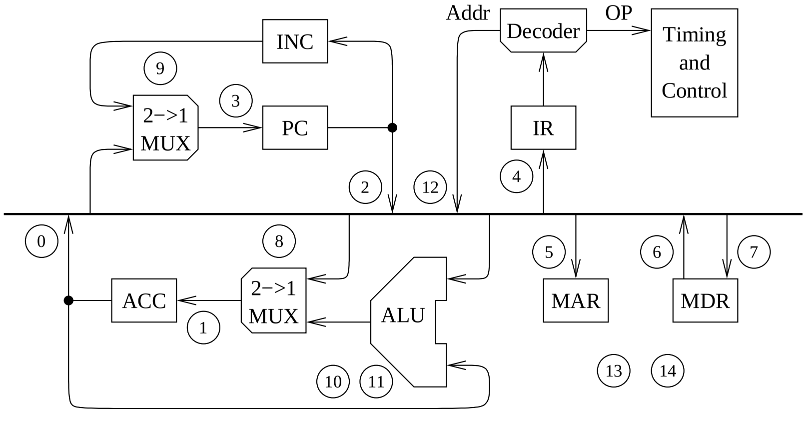
Slide 22

Assembly

10010100	LOAD	20	! $acc = a$
01010101	MPY	21	! $acc = a * b$
10111110	STORE	30	! $temp = a * b$
10010110	LOAD	22	! $acc = c$
01010111	MPY	23	! $acc = c * d$
00011110	ADD	30	! $acc = a * b + c * d$
10110110	STORE	22	! $c = a * b + c * d$

Slide 23

Data Paths and Control Points



Slide 24

Registers and Combinational Circuits

- registers
  - ACC: accumulator
  - MAR: memory address register
  - MDR: memory data register
  - PC: program counter, aka instruction counter
  - IR: instruction register
- combinational circuits
  - ALU: arithmetic and logic unit
  - decode: instruction decoder
  - INC: incrementer
  - MUX: multiplexer
  - timing and control

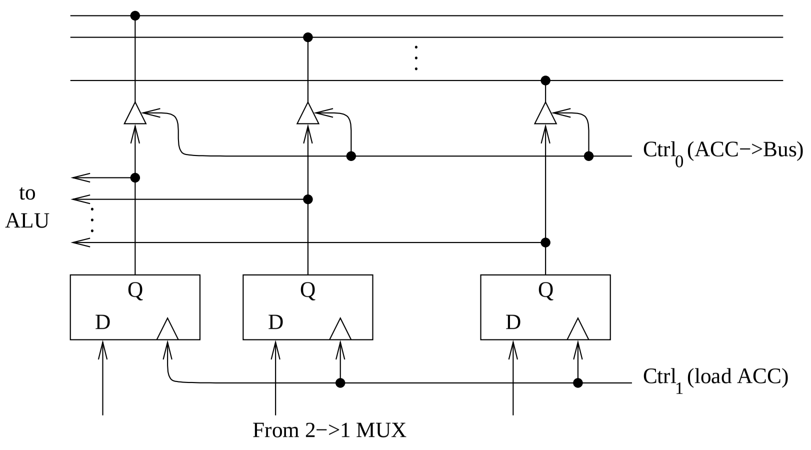
Slide 25

Control Points

Number	Operation	Number	Operation
0	ACC→bus	8	ALU→ACC
1	load ACC	9	INC→PC
2	PC→bus	10	ALU operation
3	load PC	11	ALU operation
4	load IR	12	Addr→bus
5	load MAR	13	CS
6	MDR→bus	14	$R/\overline{W}$
7	load MDR		

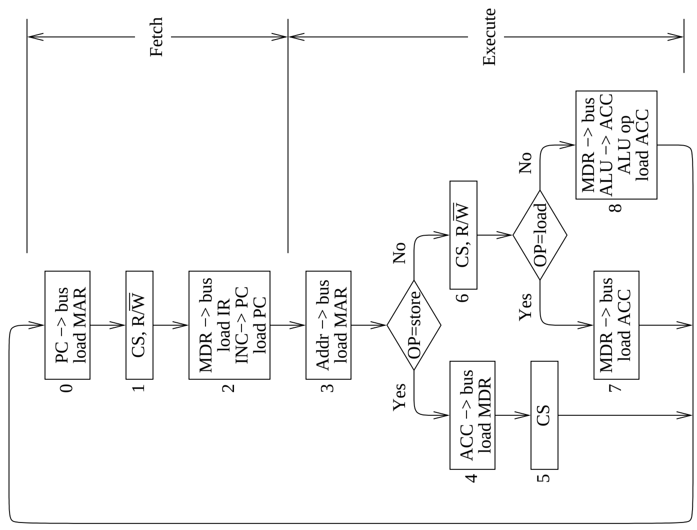
Slide 26

Controlling the Accumulator



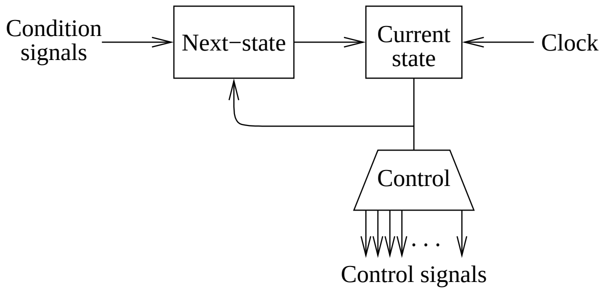
Slide 27

The ifetch Loop



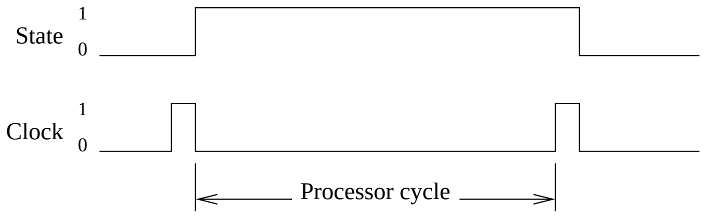
Slide 28

State Machine



Slide 29

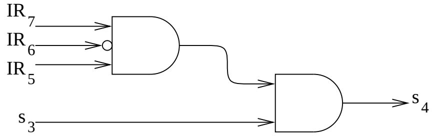
Control Signal Timing



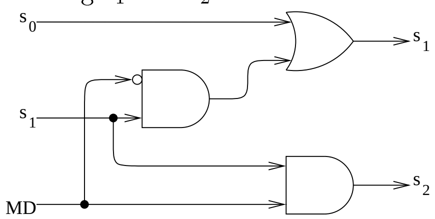
Slide 30

Generating Control Signals

- generating s_4



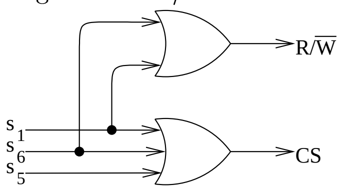
- generating s_1 and s_2



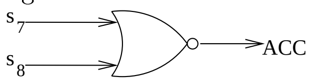
Slide 31

Generating Control Signals

- generating CS and R/$\overline{\text{W}}$

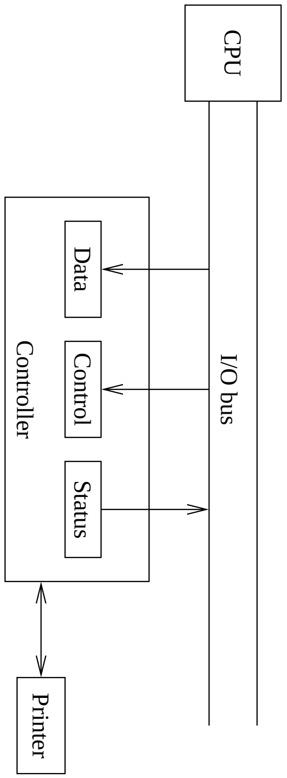


- generating load ACC



Slide 32

I/O DEVICES



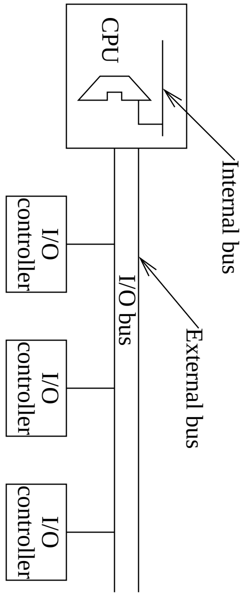
Slide 33

I/O Devices—Terminology

- *I/O Port*: the set of registers defined by an I/O controller
- *memory-mapped I/O*: the I/O ports are mapped into memory addresses, i.e., they are accessed using the memory read and write operations
- *isolated I/O*: special I/O operations for accessing the I/O ports, i.e., a separate address space

Slide 34

Buses



Slide 35

I/O Devices—Terminology

- *bus transaction*: a data transfer involving an external bus
- *(transaction) initiator*: the component that initiates the bus transaction, aka *master*
- *(transaction) initiator*: the component that responds to the bus transaction, aka *slave*
- *burst mode transaction*: a transaction that transfers several values in a single bus transaction

Slide 36

Chapter Summary

- memory
  - memory operations
  - byte ordering (big and little endian)
  - L1 and L2 caches, cache hit rate, effective access time
- the CPU
  - instruction set
  - registers, combinational circuits, control points, and buses
  - state machine implementation
- I/O devices
  - registers (control, data, and status)
  - memory mapped I/O
  - isolated I/O
- buses

  internal, external, bus transactions