Chapter 3
Components

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

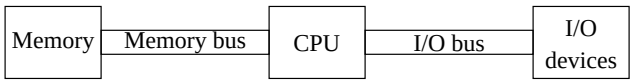
Overview

Slide 1

- Component overview
- Memory
  - operations
  - big and little endian
  - the memory hierarchy—caching
- CPU
  - an accumulator machine
  - instruction set
  - state machine implementation
- I/O devices: control, data, and status registers
- Buses: internal and external

Basic Components

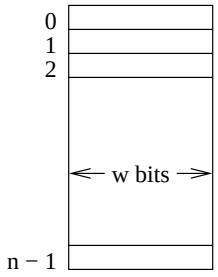
Slide 2



Basic Memory Characteristics

- Memory: a collection of independent storage units, called “cells”
- *Width*: the number of bits in each cell
- Each cell has a unique *address* (e.g., $[0, n)$)
- An $n \times w$ memory

Slide 3



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

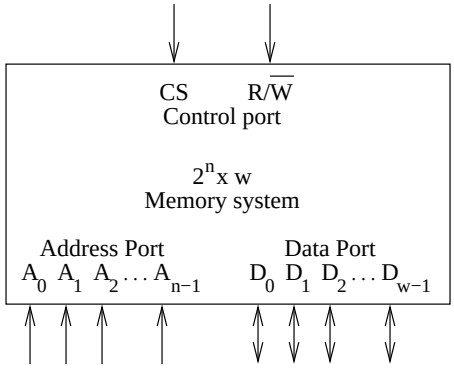
Memory Operations

Slide 4

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

Memory Ports

Slide 5



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

Steps in the Memory operations

Slide 6

- Steps in a read operation
  1. Put the address value on $A_0 \ldots A_{n-1}$; 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 address value on $A_0 \ldots A_{n-1}$; data value on $D_0 \ldots D_{w-1}$.
  2. Raise CS.
  3. Wait.
  4. Drop CS.
- *Access time*: time from the start of a read until the data is available
- *Cycle time*: time between consecutive memory operations

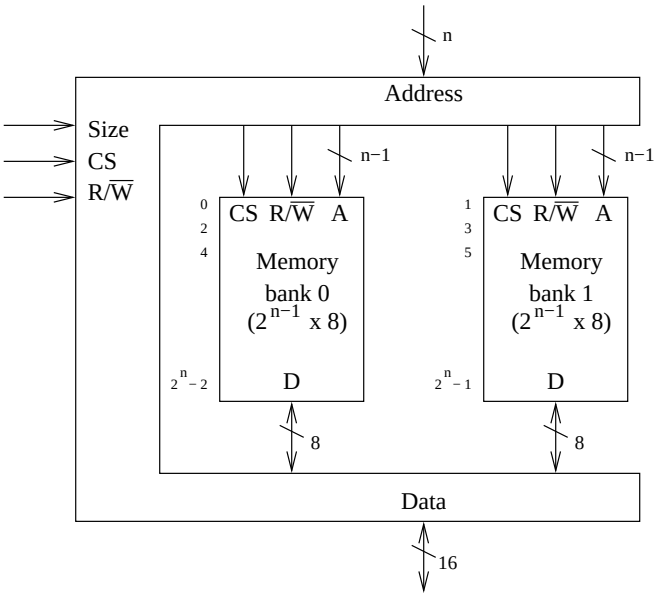
Memory Units (Bytes and Words)

Slide 7

- *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

Memory Banks

Slide 8



Terminology

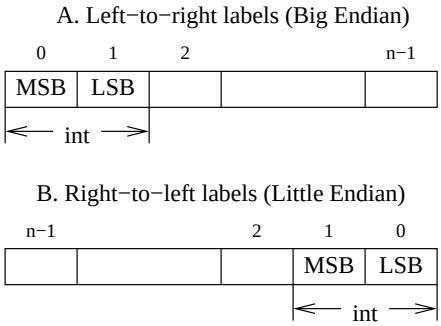
Slide 9

- *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

Byte Ordering

Slide 10

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



- Consistency: bit numbering is the same as byte numbering

The Memory Hierarchy

Slide 11

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

    50MHz means 20nsec / processor cycle

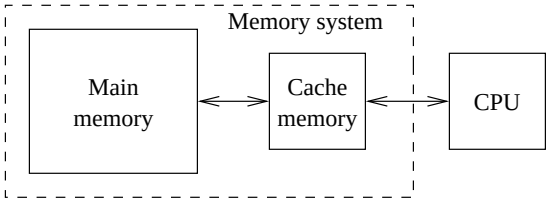
    $70/20 = 3.5$ processor cycles / memory access
- Example:
  - characteristics
    - 200MHz processor
    - 50nsec memory access
  - calculate processor cycles / memory access

    200MHz means 5nsec / processor cycle

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

Slide 12

A Single Level Cache



Cache Hit Rate and Effective Memory Access Time

Slide 13

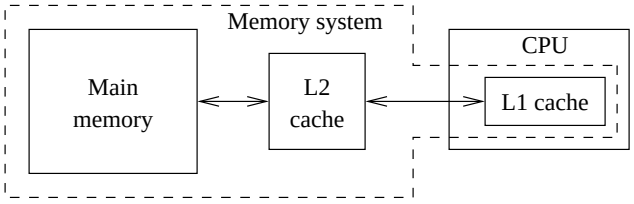
- *Cache hit rate*: the percentage of time that memory accesses are satisfied by a cache
- *Effective memory access time*: the time that it takes to access a memory value given the cache hit rate
- Example:
  - characteristics
    - 50nsec memory access time
    - 10nsec cache access time
    - 80% cache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

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

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

Slide 14

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

Slide 15

  - 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

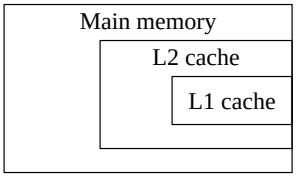
$$.80 \times 5 + .16 \times 10 + .04 \times 50 = 4 + 1.6 + 2 = 7.6\text{nsec}$$

- Same calculation without an L2 cache

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

The Memory Hierarchy

Slide 16



The CPU

Slide 17

- Data manipulation
- Running example: an accumulator machine
- Instruction set (architecture)
- Implementation (organization)
  - registers and data paths
  - instruction interpretation

An Example Calculation

Slide 18

- 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

An Instruction Set

Slide 19

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*.

An Example

Slide 20

- The calculation

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

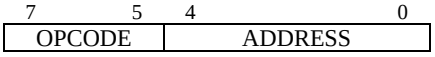
- Assume a, b, c, and d are stored in locations 20, 21, 22, and 23
- The code

```
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
```

Instruction Encoding

Slide 21

- Format



- Opcodes

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

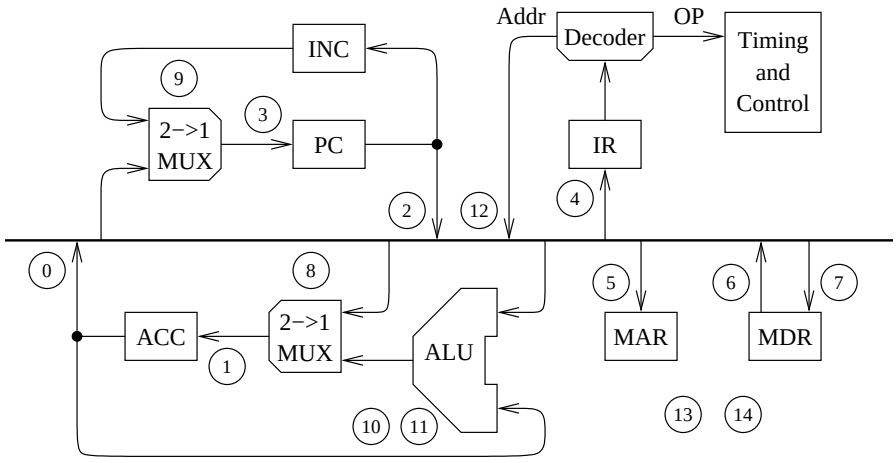
Slide 22

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



Registers and Combinational Circuits

Slide 24

- 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

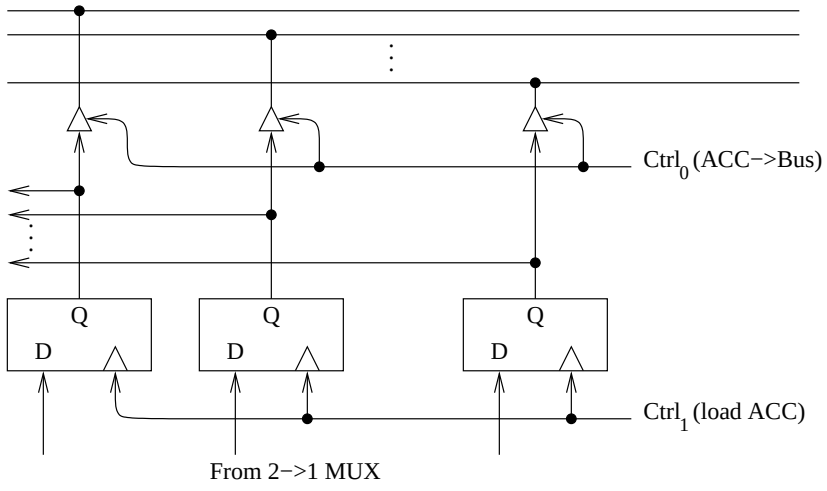
Control Points

Slide 25

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		

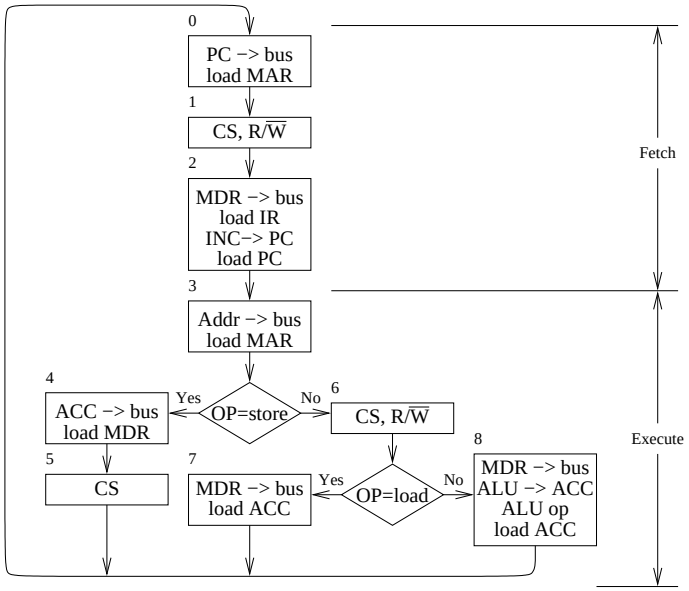
Controlling the Accumulator

Slide 26

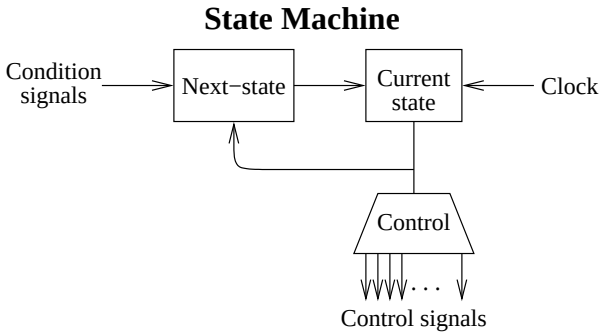


The ifetch Loop

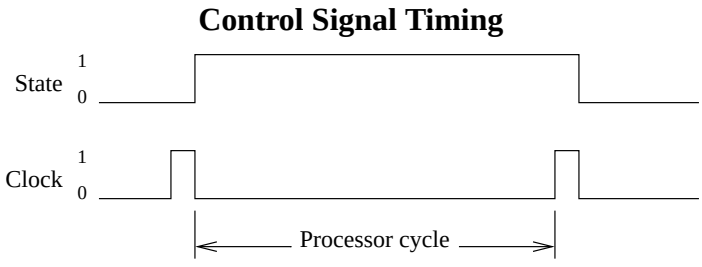
Slide 27



Slide 28



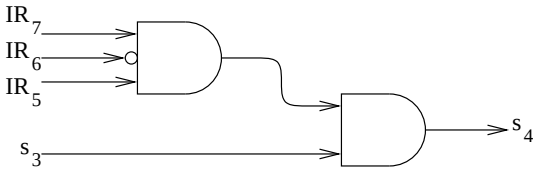
Slide 29



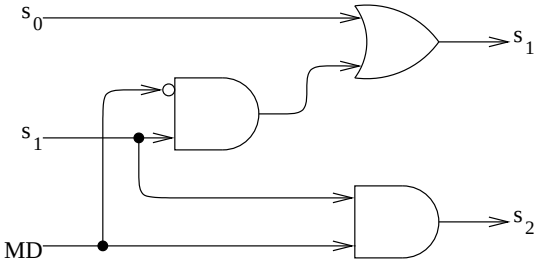
Generating Control Signals

Slide 30

- Generating s_4



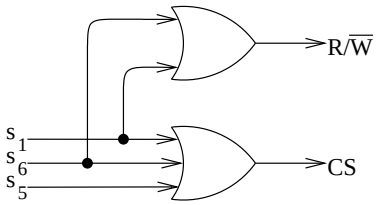
- Generating s_1 and s_2



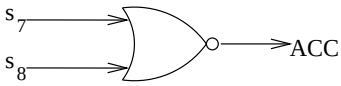
Generating Control Signals

Slide 31

- Generating CS and R/$\overline{W}$

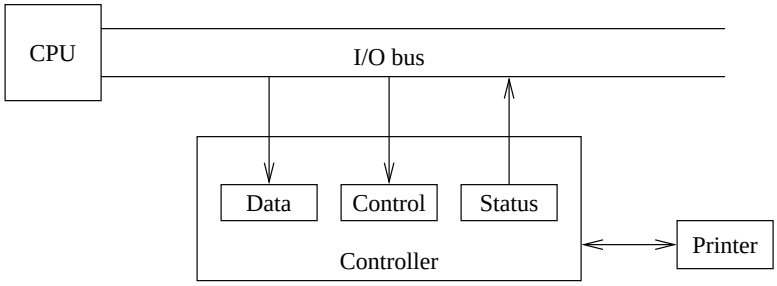


- Generating load ACC



I/O DEVICES

Slide 32



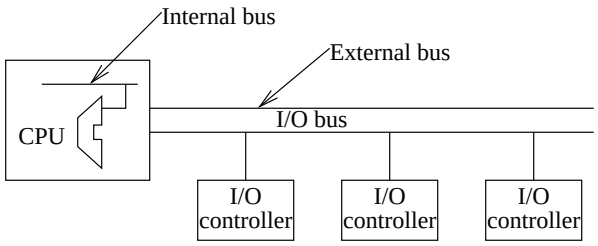
I/O Devices—Terminology

Slide 33

- *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 responder*: the component that responds to the bus transaction, aka *slave*
- *Burst mode transaction*: a transaction that transfers several values in a single bus transaction

Chapter Summary

Slide 36

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