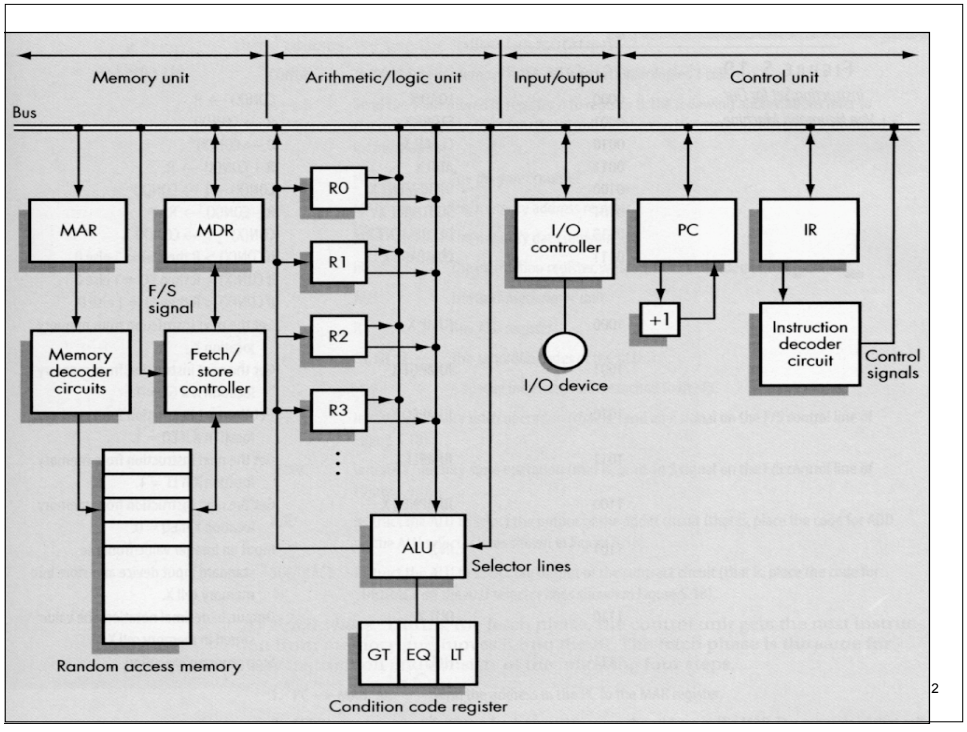
Computer Architecture (part 2)

Topics:

- Machine Organization
- Machine Cycle
- Program Execution
- Machine Language
- Types of Memory & Access



Arithmetic Logic Unit (ALU)

The ALU (Arithmetic/Logic Unit)

Circuits for performing….
mathematical operations (+, -, x, /, …)

logic operations (=, <, >, and, or, not, ...)

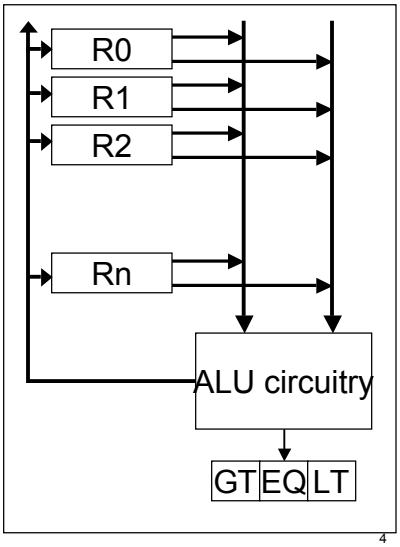
Registers

Buses connecting CPU registers and ALU registers

3

Structure of the ALU

- Registers:
  - Very fast local memory cells, that store operands of operations and intermediate results.
  - CCR (condition code register), a special purpose register that stores the result of <, = , > operations
- ALU circuitry:
  - Contains an array of circuits to do mathematical/logic operations.



4

A concrete example traced through the architecture's datapath

Question: What happens to execute this pseudocode ?

Set bankbalance to bankbalance + deposit

Needed:

a particular machine instruction set

5

Instruction Set for Textbook's Simplified Von Neumann Machine

Opcode	Operation	Meaning
0000	**LOAD X**	**CON(X) --> R**
0001	**STORE X**	**R --> CON(X)**
0010	**CLEAR X**	**0 --> CON(X)**
0011	**ADD X**	**R + CON(X) --> R**
0100	**INCREMENT X**	**CON(X) + 1 --> CON(X)**
0101	**SUBTRACT X**	**R - CON(X) --> R**
0101	**DECREMENT X**	**CON(X) - 1 --> CON(X)**
0111	**COMPARE X**	**If CON(X) > R then GT = 1 else 0** **If CON(X) = R then EQ = 1 else 0** **If CON(X) < R then LT = 1 else 0**
1000	**JUMP X**	**Get next instruction from memory location X**
1001	**JUMPGT X**	**Get next instruction from memory loc. X if GT=1**
...	**JUMPxx X**	**xx = LT / EQ / NEQ**
1101	**IN X**	**Input an integer value and store in X**
1110	**OUT X**	**Output, in decimal notation, content of mem. loc. X**
1111	**HALT**	**Stop program execution**

6

Stepping through the Machine Cycle: Adding two numbers

(To simplify, we use decimal notation rather than binary for addresses and contents)

Set bankbalance to bankbalance plus deposit

variable names	memory cell address	cell contents
bankbalance	100	500
deposit	200	250

7

Adding 2 numbers...

Machine level instructions (using textbook's machine lang.)

load content of memory location 100 to register R0

add contents of memory location 204 to register R0

store the contents of R0 in memory location 100

Memory Cell	*Cell contents*
34	Load 100
36	Add 204
38	Store 100

8

Program Counter

34

Address	*Contents*
34	Load 100
36	Add 204
38	Store 100
100	500
204	250

Fetch

PC's contents → MAR
Instruction at 34 → MDR
MDR → Instruction Reg

advance PC 36

Decode the opcode found in Instruction Register
("load", source address)

Execute

address 100 → MAR
set fetch/store controller to fetch
contents of 100 → MDR
MDR contents → R0

9

Program Counter

36

Address	*Contents*
34	Load 100
36	Add 204
38	Store 100
100	500
204	250

Fetch

PC's contents → MAR
Instruction at 36 → MDR
MDR → Instruction Reg

advance PC 38

Decode the opcode found in Instruction Register
("add", location in memory to register in ALU)

Execute

contents of register and mem location
accessed by ALU circuits
ALU circuits perform add

10

Program Counter

38

Address	*Contents*
34	Load 100
36	Add 204
38	Store 100
40	...
100	~~500~~ 750
204	250

Fetch

PC's contents → MAR
Instruction at 38 → MDR
MDR → Instruction Reg

advance PC 40

Decode the opcode found in Instruction Register ("store", CPU register, RAM address)

Execute

address 100 → MAR
set fetch/store controller to *store*
contents of R0 → MDR
MDR contents → address 100

11

Machine Instruction Set (revisited again): Instruction Set design….

8 bits	16 bits	16 bits	16 bits
Operation Code	Address Field 1	Address Field 2	Address Field 3

Hypothetical example...

OpCode	Operands	Meaning
00000101	x	add (contents at) x with contents of R0, put result back into x
00000110	x, y	add con(x) and con(y) and put result in x
00000111	x, y, z	add con(x) and con(y) and put result in z

12

Instruction Set Design: Do we need 3 ways to add?

- Reduced Instruction Set Computers (RISC)
  - Instruction set as small and simple as possible.
  - Minimizes amount of circuitry --> faster computers
- Complex Instruction Set Computers (CISC)
  - More instructions, many very complex
  - Each instruction can do more work, but requires more circuitry.

13

Machine Instruction Set (revisited): 4 classes of instructions

- Data Transfer Instructions, e.g.
  - LOAD X — Load content of memory location X to R0
  - STORE X — Load content of R0 to memory location X
  - MOVE X, Y — Copy content of memory location X to location Y

a possible RISC approach to MOVE?

14

Machine Instruction Classes (cont.)

- Arithmetic/Logic Instructions
  - ADD X, Y, Z CON(Z) = CON(X) + CON(Y)
  - AND
  - OR
- Compare Instructions
  - COMPARE X, Y
    Compare and set the condition code register (CCR)
  - If CON(X) = CON(Y) then set EQ=1, GT=0, LT=0
    - These are “condition codes” that other instructions will reference….namely...

15

Machine Instruction Classes (cont.)

- Branch Instructions - deviations from sequential control
  - JUMP X Load next instruction from memory loc. X
  - JUMPGT X Load next instruction from memory loc. X only if GT condition code is set, otherwise load statement from next sequence loc. as usual.
  - HALT

```
If (a >  b) then
  set c to a
Else
  set c to b
```

```
100   value of a
101   value of b
102   value of c
```

```
50    compare 100,101
51    jumpgt 54
52    Move 101,102
53    Jump 55
54    Move 100, 102
55    <whatever next instr. Is>
```

16

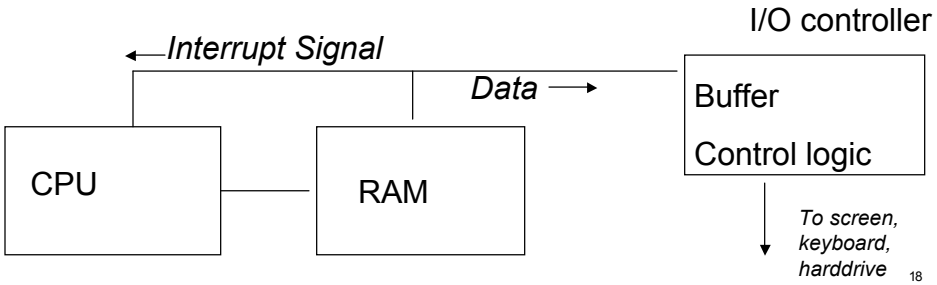
Instruction Set for Our Von Neumann Machine

Opcode	Operation	Meaning
0000	LOAD X	CON(X) --> R
0001	STORE X	R --> CON(X)
0010	CLEAR X	0 --> CON(X)
0011	ADD X	R + CON(X) --> R
0100	INCREMENT X	CON(X) + 1 --> CON(X)
0101	SUBTRACT X	R - CON(X) --> R
0101	DECREMENT X	CON(X) - 1 --> CON(X)
0111	COMPARE X	If CON(X) > R then GT = 1 else 0 If CON(X) = R then EQ = 1 else 0 If CON(X) < R then LT = 1 else 0
1000	JUMP X	Get next instruction from memory location X
1001	JUMPGT X	Get next instruction from memory loc. X if GT=1
...	JUMPxx X	xx = LT / EQ / NEQ
1101	IN X	Input an integer value and store in X
1110	OUT X	Output, in decimal notation, content of mem. loc. X
1111	HALT	Stop program execution

17

The rest of the architecture

- Input/Output
  - Access to keyboard, screen, and secondary memory
  - Issue
    - These devices are “remote” to the CPU
    - A mismatch of speed in manipulating information



18

Check point…practice problems on p. 200

19

Types of Memory: access and retrieval

ROM - read only memory

RAM - *volatile* - retrieve no less than 1 cell

- each cell has unique address
- *each cell accessed at same speed*

Secondary Memory (hard drive, CDs, removable disks)

- *non-volatile*
- each memory location has unique address
- retrieve no less than 1 memory location
- *different access times for memory locations*

20

Direct Access Memory Organization

Track + Sector Layout

tracks: concentric circles divided into sectors

sector: a block holding address plus data cells

Address for data location: a track # plus a sector #

Access to a data location is a *mechanical process*

21

Quantifying Data Retrieval Time for Direct Access Memory

Retrieval time = access time + transfer time

Recall address is a particular track and particular sector on that track

Access time = seek time + latency (rotational delay)

Transfer time:

time for entire sector to pass under read/write head

22

Example

Rotation speed: 2400 rev/min = 40 rev/sec
time for 1 revolution?
1 rev every 1/40 of a second or .025 sec
so, 25 milliseconds for 1 revolution

Arm movement: .5 msec to go to next track

tracks : 100

sectors/track: 10

	Best	worst	avg
Seek time	0	99*.5ms	
Rotational delay(latency)	0	~.025 sec	
Transfer time	1/10 (.025)	same	same

23

Practice Problems on Memory Access/Storage- pg. 189

24

Machine Architecture: Key Ideas

1. Von Neumann design

   stored program / sequential execution of instructions

2. Machine instruction set (RISC/CISC; classes of instructions; binary representation - op codes, operands)

3. Fetch/Decode/Execute Cycle

   understand the basic architecture (PC, IR, ALU, MAR, MDR, fetch/store controller)

   especially "low level movement" of bits at each stage

   review pg. 206

   role of circuits for selecting, decoding

   (multiplexors, decoders)

4. Types of Memory/ Access Operations

25

Machine Architecture: Key Skills

1. Write algorithms in machine language

   (as per practice problems)

2. Be able to reason about notions like address space, size of MAR, etc.

3. Be able to do simple calculations on accessing direct access memory

   (as per practice problems)

4. Be able to label/trace through the data-path on our simplified architecture

26