

Moon-3

CS31

Pascal Van Hentenryck



Overview

Moon-3

- Multiple Cycle Bus-based architecture of Moon
- Bus organization
- More on breaking instructions
- More on control

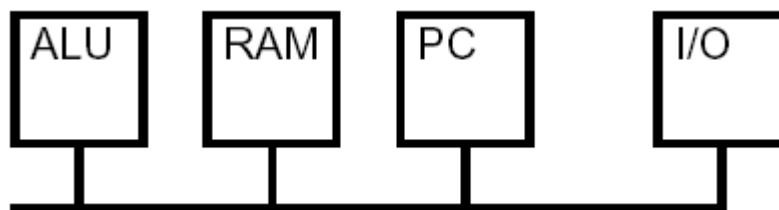
Bus Architecture

Basic Observation

- In a multiple-cycle architecture, a component can be used several times during an instruction (e.g. RAM)
- Possibility to reduce the hardware

Bus Architecture

- Use of a bus to reduce the wires between the various components since many of them communicate with each other
- Widely used for most computers before the 80s, when cost has to be minimized (and traded off for speed)



Bus Architecture

What can be shared in Moon-2?

- The ram can be used to store both the program and the data

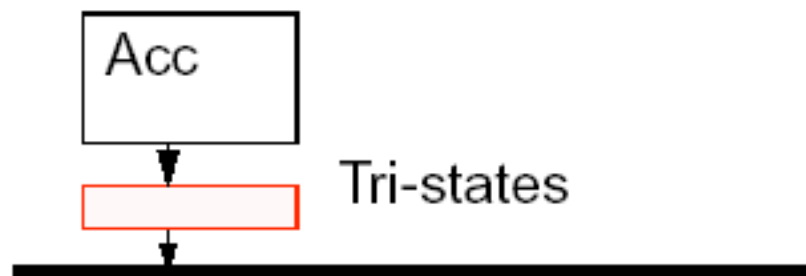
Basic Idea

- Design a version of Moon where there is a single memory unit
- There are many components that interact with the ram now.

Bus Architecture

What is the difficulty?

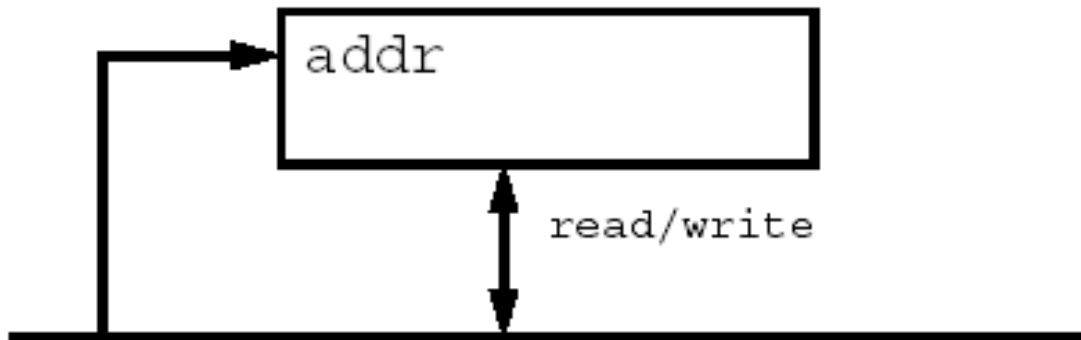
- must make sure that the bus is driven by at most one input at any time.
- use of tri-state devices (remember them?)



- need to break the instructions into smaller pieces

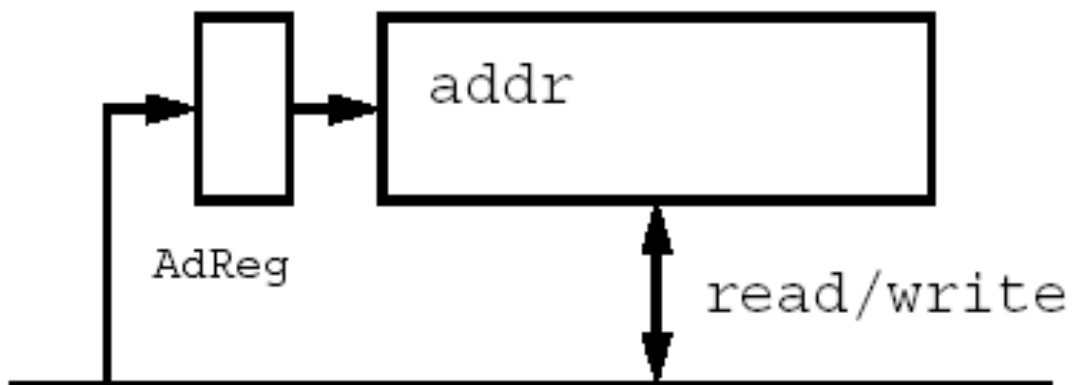
Bus Architecture

Typical Example



Problem

- cannot put the address on the bus (from IR) and the result of a read/write at the same time
- Need of an address register





Instruction LAD

Cycle 1 (Fetch)

`addr := PC;`

Cycle 2 (Fetch)

`Ir := ram[addr];`

Cycle 3 (execute)

`addr := Ir;`

Cycle 4 (execute)

`Acc := ram[addr];`

`Pc := Pc + 1;`

Instruction LIM

Cycle 1 (Fetch)

`addr := PC;`

Cycle 2 (Fetch)

`IR := ram[addr];`

Cycle 3 (execute)

`ACC := IR;`

`PC := PC + 1;`

Cycle 4 (execute)

Instruction SAD

Cycle 1 (Fetch)

addr := PC;

Cycle 2 (Fetch)

IR := RAM[addr];

Cycle 3 (execute)

addr := IR;

Cycle 4 (execute)

RAM[addr] := Acc;

PC := PC + 1;

Instruction SOU

Cycle 1 (Fetch)

addr := PC;

Cycle 2 (Fetch)

IR := ram[addr];

Cycle 3 (execute)

out := Acc;

PC := PC + 1;

Cycle 4 (execute)

Instruction SUB

Cycle 1 (Fetch)

addr := PC;

Cycle 2 (Fetch)

IR := RAM[addr];

Cycle 3 (execute)

addr := IR

Cycle 4 (execute)

AR := RAM[addr];

Cycle 5 (execute)

Acc := ACC - AR;

PC := PC + 1;

Instruction JZ

Cycle 1 (Fetch)

addr := PC;

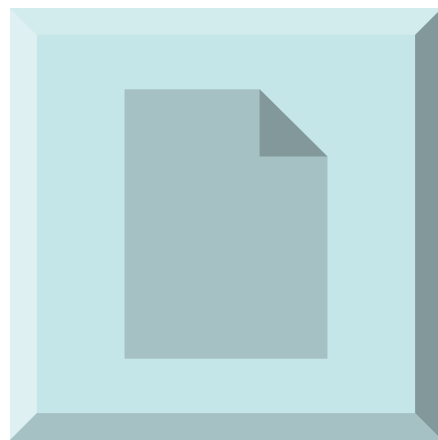
Cycle 2 (Fetch)

- IR := RAM[addr];

Cycle 3 (execute)

- PC := if ACC = 0 then IR
else PC + 1;

Cycle 4 (execute)



Instruction LAD

	1	2	3	4	5
PcW					
BcPc					
MPc					
AcW					
BcAc					
MAc					
AdrW					
OutW					
Read					
Ce					
IrW					
IrBc					
ArW					
NS2					
NS1					
NS0					

Instruction LIM

	1	2	3	4	5
PcW					
BcPc					
MPc					
AcW					
BcAc					
MAc					
AdrW					
OutW					
Read					
Ce					
IrW					
IrBc					
ArW					
NS2					
NS1					
NS0					

Instruction SAD

	1	2	3	4	5
PcW				1	
BcPc					
MPc					
AcW					
BcAc				1	
MAc					
AdrW			1		
OutW					
Read					
Ce				1	
IrW					
IrBc			1		
ArW					
NS2					
NS1					
NS0					

Instruction SUB

	1	2	3	4	5
PcW					1
BcPc	1				
MPc					
AcW					1
BcAc					
MAc					1
AdrW	1		1		
OutW					
Read		1		1	
Ce					
IrW		1			
IrBc			1		
ArW				1	
NS2				1	1
NS1		1	1		
NS0	1		1		1