

Moon-2 (II)

CS31

Pascal Van Hentenryck



Overview

Moon-2 (II)

- Control
- State-Based Controllers

Controlling Moon-2

What needs to be controlled?

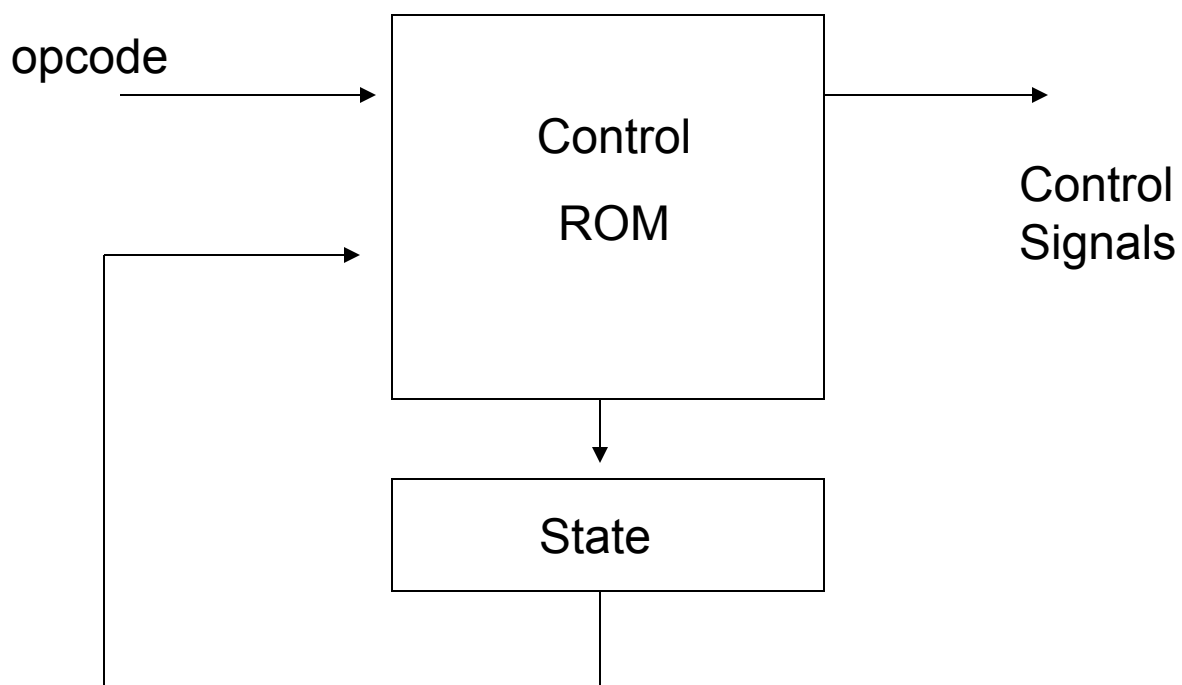
- PcWrite
- MuxPc
- IrWrite
- RamWrite
- MuxIm
- MuxMem
- AccWrite
- ArWrite
- OutWirte

Controlling Moon-2

How to control it?

- opcodes
- cycle of the instruction

State based controller



Controlling Moon-2

How many bits to represent the state?

- 2 since an instruction takes at most 3 cycles

What size is needed for the ROM?

- Inputs: 5 bits (3 for opcode and 2 for state)
- Outputs: 11 bits (9 control signals)

Instruction LAD

LAD	C1	C2
PcW	0	1
MPc	0	0
IrW	1	0
RaW	0	0
MIm	0	0
Mme	0	1
AccW	0	1
ArW	0	0
OutW	0	0
Ns1	0	0
Ns0	1	0

Instruction LIM

LIM	C1	C2
PcW	0	1
MPc	0	0
IrW	1	0
RaW	0	0
MIm	0	1
Mme	0	1
AccW	0	1
ArW	0	0
OutW	0	0
Ns1	0	0
Ns0	1	0

Instruction SAD

SAD	C1	C2
PcW		1
MPc		0
IrW		0
RaW		1
MIm		x
Mme		x
AccW		0
ArW		0
OutW		0
Ns1		0
Ns0		0

Instruction SOU

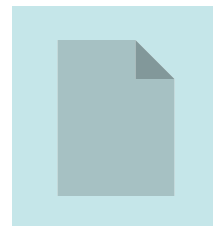
SOU	C1	C2
PcW		1
MPc		0
IrW		0
RaW		0
MIm		0
Mme		0
AccW		0
ArW		0
OutW		1
Ns1		0
Ns0		0

Instruction SUB

SUB	C1	C2	C3
PcW	0	0	1
MPc	0	0	0
IrW	1	0	0
RaW	0	0	0
MIm	0	0	0
Mme	0	0	0
AccW	0	0	1
ArW	0	1	0
OutW	0	0	0
Ns1	0	1	0
Ns0	1	0	0

Instruction JZ

JZ	C1	C2
PcW		1
MPc		1
IrW		0
RaW		0
MIm		0
Mme		0
AccW		0
ArW		0
OutW		0
Ns1		0
Ns0		0



Types of Machines

Accumulator Machines (Moon)

- Accumulator stores results
- Arithmetic operations uses one operand and the accumulator
- Load from memory to the accumulator
- Store in memory to the accumulator
- Instructions have one address.

Why the accumulator?

Types of Machines

Stack machines

- Stack to replace the accumulator
- Arithmetic operations only use the stack
- Load to memory to the stack
- Store into memory from the stack

Examples

- Java virtual machine

Why the stack?

Types of Machines

Stack Machine Assembly

push a

push RAM[a] on the stack

iload v

push v on the stack

pop a

pop the top of the stack in RAM[a]

add

pop (x,y) from the stack and push
 $x + y$ on the stack

Types of Machines

Accumulator vs Stack Machines

- evaluate $(a1*a2)+(a3*a4)$

Stack

```
push a1
push a2
mult
push a3
push a4
mult
add
```

Accumulator

```
load a1
mult a2
store a1000
load a3
mult a4
add a1000
```

Types of Machines

Register Machines

- fixed (but reasonable) number of registers
- Instructions apply on registers and on memory or both
 - `add r1, a1, r2`
- Registers are general-purpose

Examples

- MIPS, modern machines

Types of Machines

Register vs Stack Machines

- evaluate $(a1*a2)+(a3*a4)$

Stack

```
push a1
push a2
mult
push a3
push a4
mult
add
```

Registers

```
load r1, a1
load r2, a2
mult r1, r2, r3
load r2, a3
load r3, a4
mult r2, r2, r3
add r1, r1, r2
```