Lecture 9 - Components and Instructions

What are the components

Registers

These are high speed storage for very small amounts of data - like 16 bits. So it is memory with speed. Often in other machines you see a “register-file” where you have a bunch of registers that you can address. In our design we have some specific registers.

1. PC : Program Counter.
2. IR : Instruction register
3. Results : Output from a computation
4. AC : Accumulator - one of 2 operands for the computation.
5. MAR : Memory Address Register
6. MDR : Memory Data Register (memory buffer register)
7. Input: A means of getting input into the machine
8. Output: A means of getting input into the machine

Storage - 2 kinds

Main Memory

Microcode Memory

Microcode PC - how microcode counts up.

Mux

A mux is a way to switch between different inputs. Our design uses a 4 input 8 wide mux. This means 2 control lines. Values are 00, 01, 10, 11 for the control.

We use this with outputs from the Microcode to perform jumps in the microcode control and to decode the instructions. This is also how we jump based on outputs from result register.

Microcode Control

Microcode is just a turn on / off memory that we use to control the chunks of the system. Instead of microcode you could do this as hard-coded. Most CPUs today are Microcode driven. This is how we can have updates to the Intel hardware after it is in the field.

ALU

The computational capability of the system is the Arithmetic Logic Unit (ALU). This is where all the instructions actually get turned into an action.

Our ALU has 4 control inputs.

i3	i2	i1	i0	Used	Action Taken
0	0	0	0	*	2s Compliment
0	0	0	1		
0	0	1	0	*	Increment by 1, ac + 1 -> Result
0	0	1	1		Decrement by 1, 2s compliment, result = ac - 1
0	1	0	0	*	Add: result = ac + bus (mdr usually)
0	1	0	1	*	Sub: subtract A - B
0	1	1	0		A >> B - Arithmetic - fills with MSB
0	1	1	1	*	A == B - if A == B, result <- 1
1	0	0	0		Compliment: Toggle each bit in result = ^ac
1	0	0	1	*	1 if AC less than 0, 2s compliment
1	0	1	0	*	1 if AC greater than 0, 2s compliment
1	0	1	1		A and B
1	1	0	0		A or B
1	1	0	1		A xor B
1	1	1	0		A >> B - logical - 0 fill - Shift Right
1	1	1	1		A << B - logical - 0 fill - Shift Left

Logic

Our logic for this system is a tiny bit that connects between the instruction register and the mux. Most systems refer to this as instruction decoding.

Example - Take the first 4 bits of the instruction - add to it a few bits from the microcode - and - jump to a location in the microcode.

Cache etc.

Fancier systems have multiple layers of caching to make things faster.

More on instructions

Instructions

Last time: Load / LoadI

This time

1. Add
2. Clear AC
3. Store
4. How to Increment a Value
5. Input
6. Output

An Emulator for Instructions