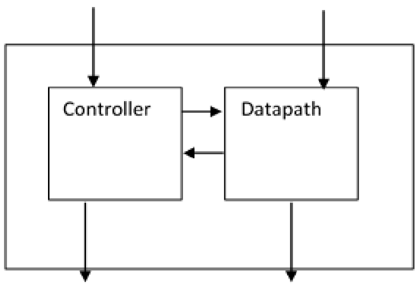
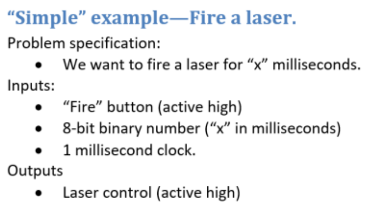
Today:

- Bus-based Datapath
  - Find Max
- Generic computer

Background

We've done some basic datapath and control work. We started with a simple device to fire a laser, in homework you've worked on a device that reads 16 values from memory and averages them, and in both lecture and homework you've worked on a 4-bit multiplier.





"Simple" example—Fire a laser.

Problem specification:

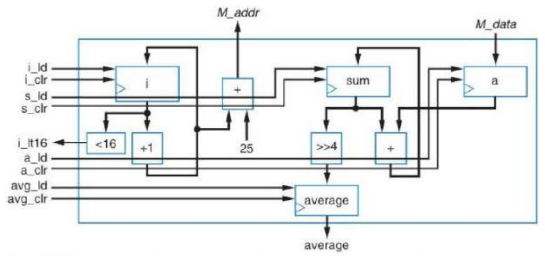
- We want to fire a laser for "x" milliseconds.

Inputs:

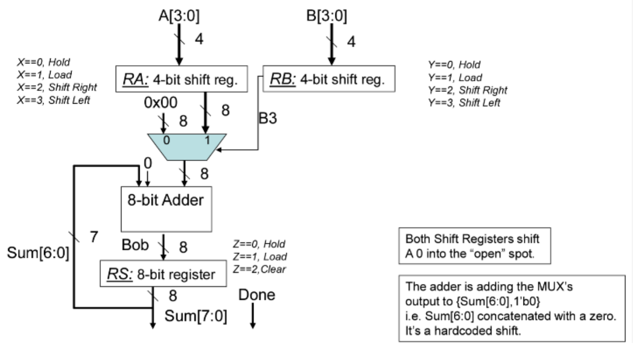
- "Fire" button (active high)
- 8-bit binary number ("x" in milliseconds)
- 1 millisecond clock.

Outputs

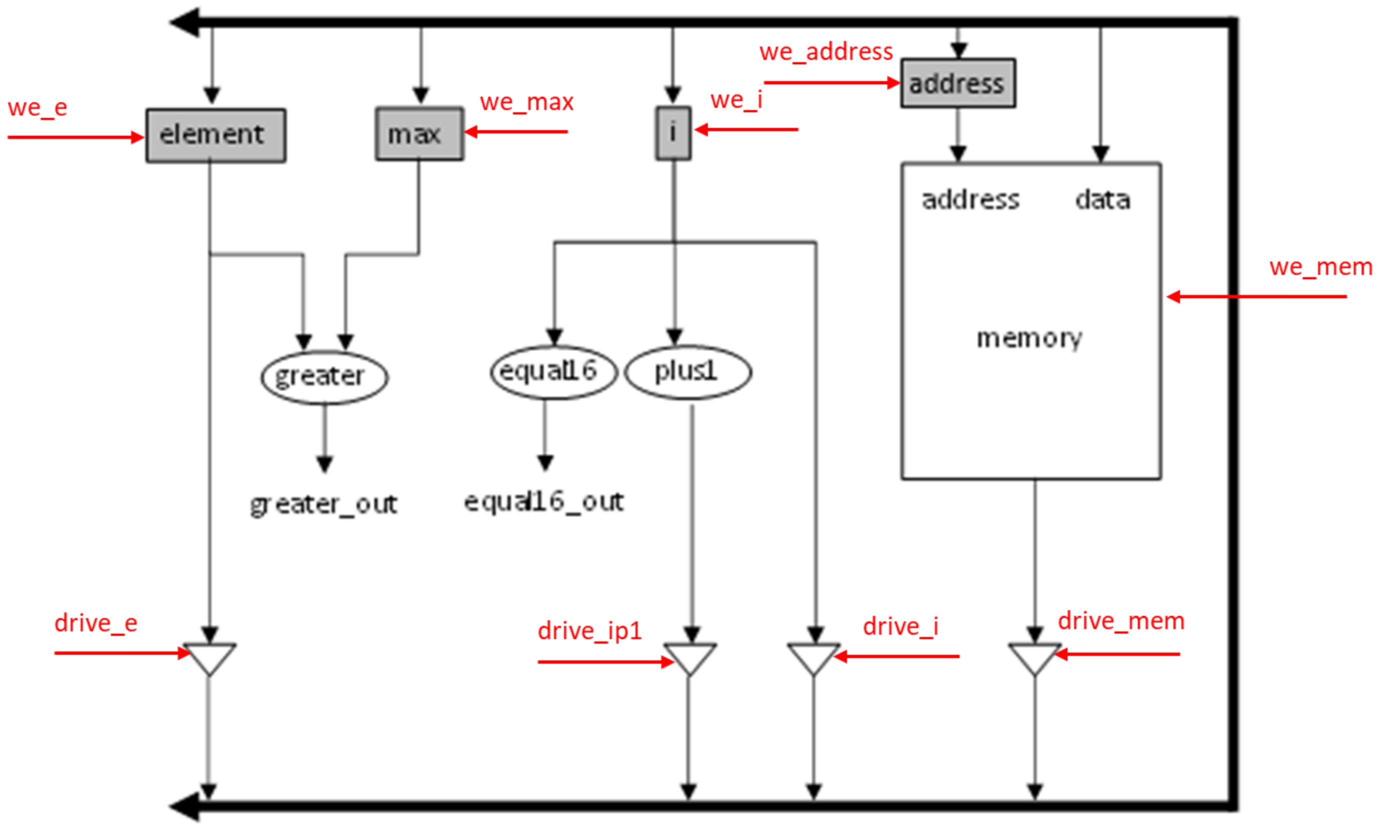
- Laser control (active high)



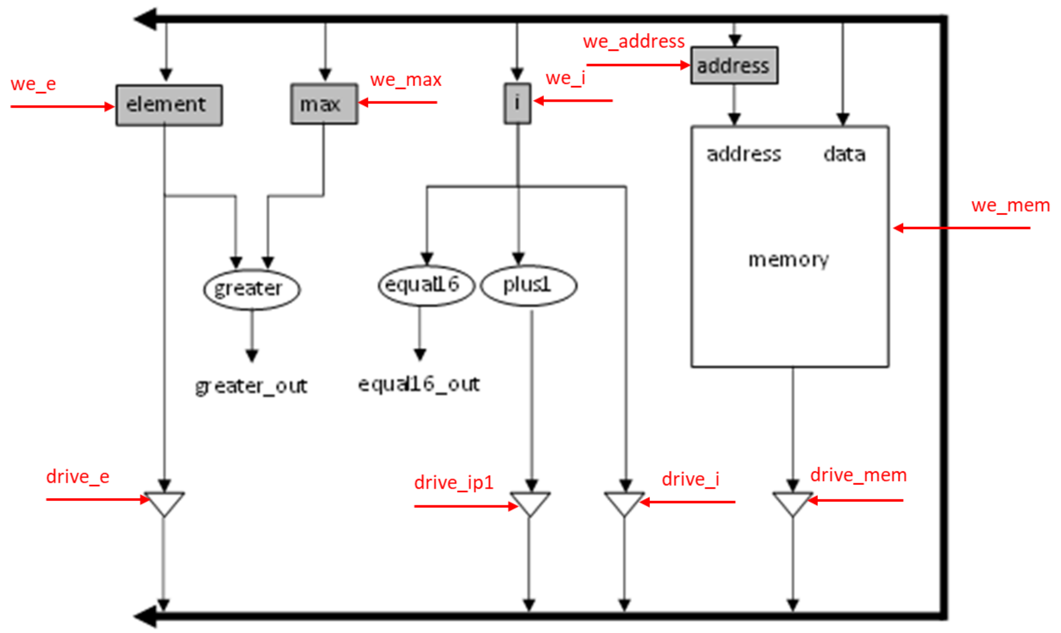
In each case we had a datapath (usually given) and a control part. Lab 6 could even be viewed in a similar way—the counter is the (fairly simple) datapath.



Today we'll look at bus-based datapath schemes including a simple computer. This gets much closer to "RTL" (Register transfer logic) in that you can think of each state as defining how data moves between registers (and memories). We're just going to look at systems with a single bus—they can be a bit easier to think about. Notice that the datapaths we've already looked at have more than one bus—they directly connect things. This is quite reasonable—maybe better in practice. But we're going to be doing some fairly complex things, so we're going to go with a datapath that is a bit easier to work with.



The above is a *proposed* datapath which is to search the first 16 elements of memory and put the largest value into the 17th element of memory (address 16—it's zero indexed). All of the shaded boxes are registers. Memory is, of course, an array of registers. There is one bus (it wraps around) and each cycle only one device can be "driving the bus" but multiple devices could, in theory, read from it. (That should make sense, yes?). The we_ are all "write enables". The "drive_" are all tri-state device controllers. Let's try to build a controller that does what we need to get the max of the first 16 memory locations into location 16 of the memory. You have more room on the next page…


we_e
element
max
we_max
i
we_i
we_address
address
address
data
memory
we_mem
greater
greater_out
equal16
equal16_out
plus1
drive_e
drive_ip1
drive_i
drive_mem

Start on a computer

Instruction set:

Instruction name	Opcode	Effect
halt	0	`PC = PC+4` `stop executing instructions`
add	1	`PC = PC+4` `memory[addr0] = memory[addr1] + memory[addr2]`
sub	2	`PC = PC+4` `memory[addr0] = memory[addr1] - memory[addr2]`
mult	3	`PC = PC+4` `memory[addr0] = memory[addr1] * memory[addr2]`
div	4	`PC = PC+4` `memory[addr0] = memory[addr1] / memory[addr2]`
cp	5	`PC = PC+4` `memory[addr0] = memory[addr1]`
and	6	`PC = PC+4` `memory[addr0] = memory[addr1] & memory[addr2]`
or	7	`PC = PC+4` `memory[addr0] = memory[addr1] | memory[addr2]`
not	8	`PC = PC+4` `memory[addr0] = ~memory[addr1]`
be	9	`if (memory[addr1] == memory[addr2]) {` `PC = addr0` `} else {` `PC = PC+4` `}`
bne	10	`if (memory[addr1] != memory[addr2]) {` `PC = addr0` `} else {` `PC = PC+4` `}`
blt	11	`if (memory[addr1] < memory[addr2]) {` `PC = addr0` `} else {` `PC = PC+4` `}` Comparisons take into account the sign of the number. E.g., 16'hffff (-1) is less than 16'h0000 (0).

Instructions are spread out over 4 addresses. Opcode, addr0, addr1, addr2.

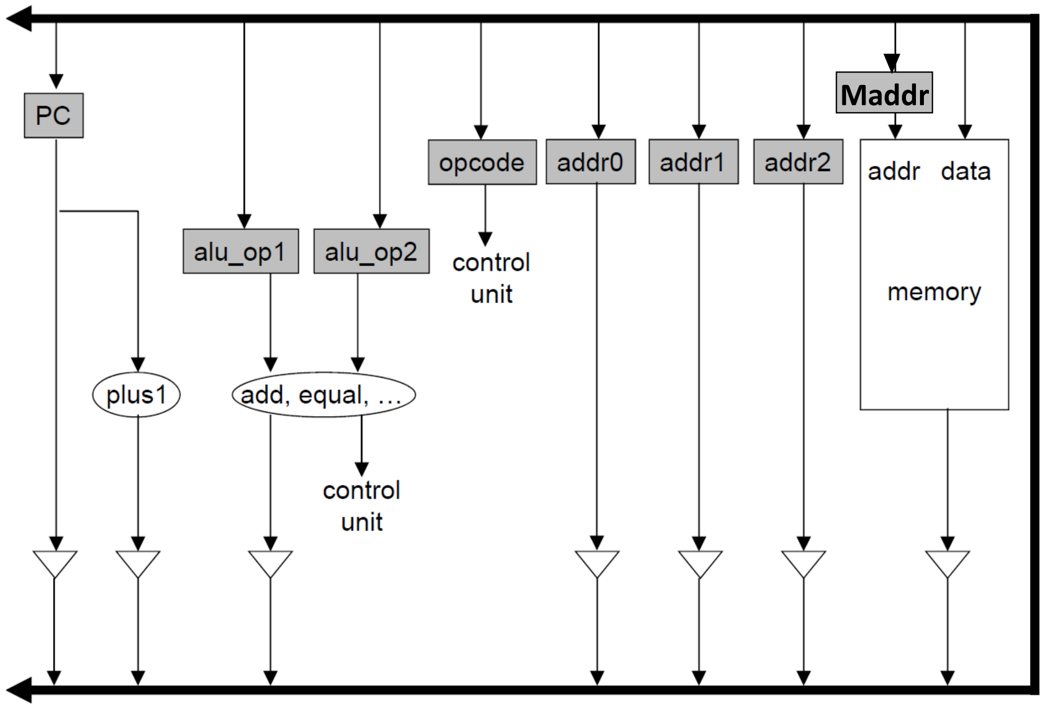
For example, if memory holds the data on the right, then what is the instruction supposed to do?

Location	Data
0	1
1	10
2	20
3	30
4	0
5	0
6	0
7	0

Write a short program that performs A=B+C+D where A is location 50, B is location 51, C is location 52 and D is location 53.

Location	Data
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	

Let's look at a datapath that can do these instructions. We'll only look at add in class, but it gives the idea.



Control points:

PC_drive, plus1_drive, ALU_drive, addr0_drive, addr1_drive, addr2_drive, memory_drive.

PC_en, op1_en, op2_en, opcode_en, addr0_en, addr1_en, addr2_en, Maddr_en, memwrite_en

Draw a state machine for add. What part would be in common with the other instructions?