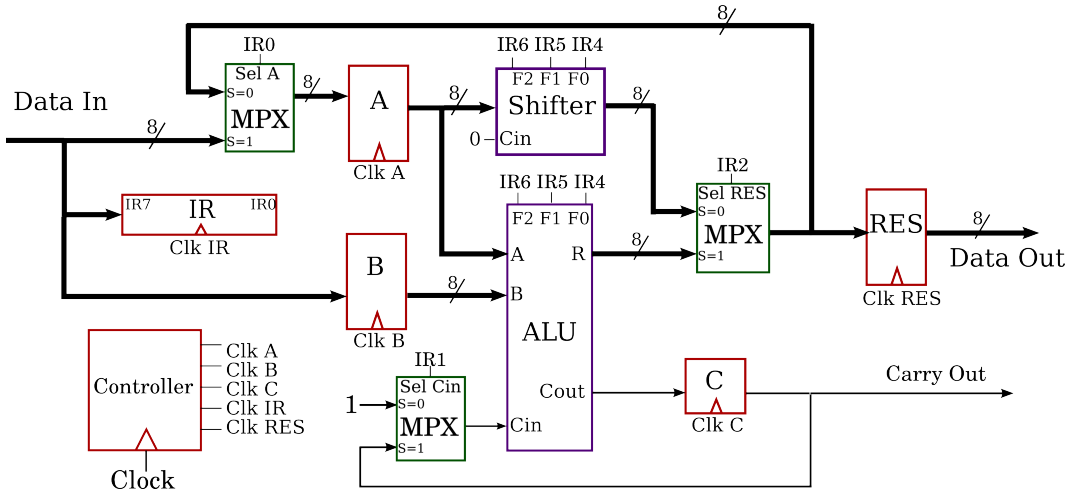
Tutorial 9: The Manual Processor

In lectures 14 and 15 we developed a simple eight-bit externally programmed computer. We now examine in detail how it works.



Selection Bits	000	001	010	011	100	101	110	111
ALU Function	0	B-A	A-B	A plus B	A xor B	A+B	A.B	-1
Shifter Function	Unchanged	Rotate left	Shift left C=0	Shift left C=Cin	Rotate Right	Shift Right C=0	Shift Right Cin=I[7]	Shift Right C=Cin

The basis of all digital computer operations is the register transfer operation. When the clock is applied one or more register transfer operations take place. The function carried out by a register transfer operation is determined by the bits stored in the Instruction Register or IR. The bits in this register perform two operations:

1. Determine the functions of the arithmetic hardware - the ALU and Shifter.
2. Set the select lines of multiplexers to connect the required registers . For example, if IR0=IR2=0 the result of the shifter operation is connected back to register A. If IR0=1 the Data In bus is connected to A.

In addition, registers are updated by the application of clock signals. During the proper operation of the processor, the following clock signals are generated by the controller:

State	1	2	3	4	5
Clock	IR	A	B and C	IR	RES and C

This sequence is repeated as long as the processor is running. Now we have enough information to specify the bits in the input data which are loaded into the IR during clock times 1 and 4. For example:

ADD: Add two numbers together

The numbers to be added will be set on the Data In lines during states 2 and 3 of the processor cycle. The processor operates as follows:

State	Register transfer operations	Multiplexer and Function Selection
1	IR ← DataIn (Opcode 1)	Does not matter
2	A ← DataIn	A connected to DataIn
3	B ← DataIn and C ← 0	ALU=0 and Cout=0
4	IR ← DataIn (Opcode 2)	Does not matter
5	Res ← A pl B, C ← ALU/Cout	ALU=plus, Cin connected to C

We can specify the input data necessary to do the required instruction. Notice that the transfer takes place at the end of the state; therefore, the contents of the IR change just before states 2 and 3, as shown below.

		Instruction register (IR)							
State	Data In	IR7	IR6	IR5	IR4	IR3	IR2	IR1	IR0
1	x000xxx1	X	X	X	X	X	X	X	X
2	Number 1	X	0	0	0	X	X	X	1
3	Number 2	X	0	0	0	X	X	X	1
4	x011x11x	X	0	0	0	X	X	X	1
5	xxxxxxxx	X	0	1	1	X	1	1	X

The select line of the multiplexer which controls the ALU/Cin signal (IR1) is a ”don’t care” for steps 1 to 4 because the ALU function selection is 000 (IR6-IR4) and the ALU provides output 00000000 disregarding what the carry input to the ALU is. As shown on the diagram, this bit if set to 0 selects the logic 1 input, and if set to 1 selects the C register.

In the problems below you are asked to specify the required data inputs for the five states in order to execute some computer operations. Indicate as many ”don’t care” bits as possible since this may reduce the complexity of the outside circuit which controls this externally controlled processor.

Problem 1: INC(Increment) Result = Number + 1

The number to be incremented will be set on the Data In lines during both state 2 and state 3.

		Instruction register (IR)							
State	Data In	IR7	IR6	IR5	IR4	IR3	IR2	IR1	IR0
1									
2	Number								
3	Number								
4									
5	xxxxxxxx								

Problem 2. Result = (Number1 plus Number2) div 2

(This will require two complete five-state cycles, and you may assume that no carry is generated)

		Instruction register (IR)							
State	Data In	IR7	IR6	IR5	IR4	IR3	IR2	IR1	IR0
1									
2	Number1								
3	Number2								
4									
5	xxxxxxxx								
1									
2	xxxxxxxx								
3	xxxxxxxx								
4									
5	xxxxxxxx								

Problem 3

In the current design of the processor we provided a carry register for the ALU so that we could perform arithmetic on sixteen bit (or bigger) integers. For example, the carry would be set to zero and the low order bytes would be added and the carry stored and then the high order bytes would be added with the previous carry. Can you design some hardware that would give similar functionality to the shifter?