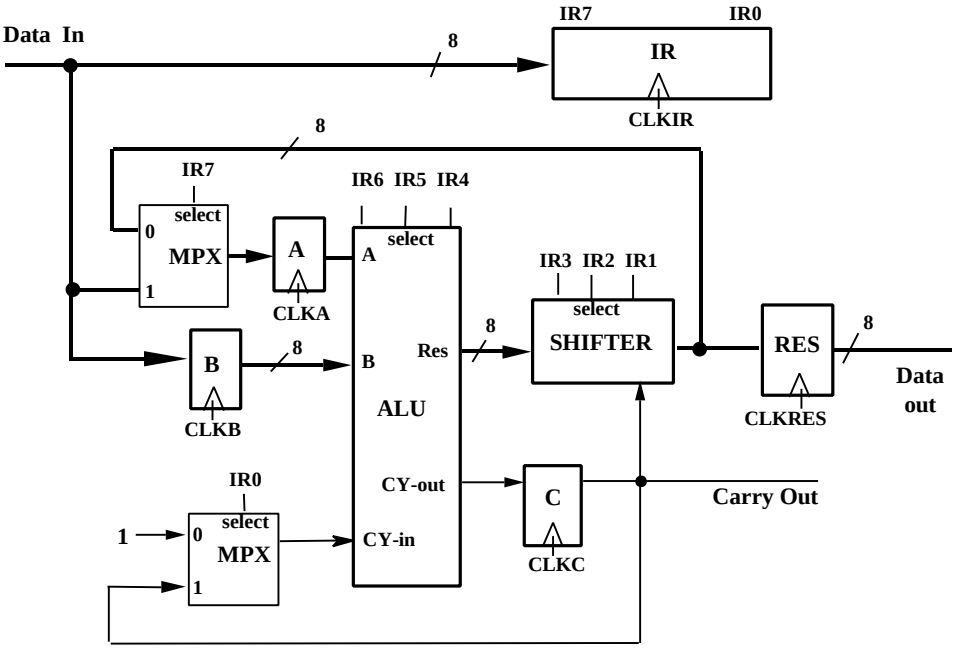
Department of Computing
Course 112 - Hardware
Tutorial 9

This tutorial contains exercises that are assessed. Make certain that your name, initials and group are clearly filled in. Do all your work on this sheet and hand it in at the end of the tutorial session.

LAST NAME_____ INITIALS_____ GROUP_____

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 output	All bits 0	B mi A	A mi B	A pl B	A xor B	A + B	A ·B	All bits 1
Shifter	unchanged	left shift Cin = 0	right shift Cin = 1	right shift Cin = 0	unchanged	left shift Cin = Carry	right shift Cin = Data[7]	right shift Cin = Carry

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 type and number of register transfer operation(s) are controlled by the bits stored in the **Instruction Register** or **IR**. The bits in this register perform the following functions:

a. Determine the functionality of controlled function generators such as the ALU and Shifter.
b. Set the select lines of multiplexers to connect the required registers . For example, if IR7 is zero the shifter is connected to register A. If it is one the data input bus is connected to A

In addition, registers are updated by the application of clock signals.

The **ALU** and **Shifter** functions are shown in the next two tables:

During the proper operation of the processor, the following clock signals are generated:

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

State	Register transfer operations	Multiplexer and Function Selection
1	IR <- DataIn (Opcode 1)	Does not matter (IR is loaded)
2	A <- DataIn	A connected to DataIn (A is loaded)
3	B <- DataIn C <- 0	ALURes and CYout=0 (B,C are loaded)
4	IR<- DataIn (OpCode 2)	Does not matter (IR is loaded)
5	Res <-A pl B and C <- Carry of (A pl B)	ALU=plus, Shifter unchanged, ALUCyin connected to C

Thus, 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.

State	Data In								Instruction Register (IR)							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1	1	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X
2	Number 1								1	0	0	0	X	X	X	X
3	Number 2								1	0	0	0	X	X	X	X
4	X	0	1	1	0	0	0	1	1	0	0	0	X	X	X	X
5	X	X	X	X	X	X	X	X	X	0	1	1	0	0	0	1

The select line of the multiplexer which controls the **ALU/CYin** signal **(IR0)** 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 set to **0** selects the logic **1** input, set to **1** selects the the **C** input. You are asked to specify the required data inputs for the five states in order to execute a given computer operation. 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.

State	Data In								Instruction Register (IR)							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1																
2																
3																
4																
5																

Problem 2.

FN1 (A+B-C)/2 Result = (Number1+Number2-Number3)/2

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

State	Data In								Instruction Register (IR)							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1																
2																
3																
4																
5																
State	Data In								Instruction Register (IR)							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1																
2																
3																
4																
5																

Optional Problem

Many complex instructions would require the loading of the **B** register from the **A** register. Redesign the processor hardware such that this becomes possible.