Introduction to Digital Logic
EECS/CSE 31L

Assignment 2 Design Report
Designing Basic Processor

Prepared by: Sunny Xu
Student ID: 60905232

EECS Department
Henry Samueli School of Engineering
University of California, Irvine

September 4, 2016

1 BLOCK Description

This processor is designed to read from a “memory” component on a positive edge and write during a negative edge on the clock cycle. The memory will return a 32-bit instruction set that will contain type (whether the operation will be read or utilizing the “immediate” portion of the instruction set), read addresses, write address, and the 4 bit function that will be performed on the designated numbers. The two addresses will be read from a register file which will contain a 2^6 – 1 size memory all initialized as. The 15 bit immediate operand will have its sign extended to 32 bits. After using a multiplexor to determine which of the “B” operand to use (either the immediate type or the second address used to read in the register file), the two 32-bit operands will be processed by the 32-bit ALU. The result from the ALU (not including the COUT output it also processes) will then be written to the designated write address in the register file.

The controller is designed to accept a clock input for its clock cycle to be controlled. The test bench can simply consists of an alternating clock cycle for this processor design to function.

Everything except for the ALU is has GENERIC implemented in the components so when instantiating them in the controller, the user has the choice to change the number of bits with ease.

All the components are contained in a package called “proc_components” which will contain everything that the controller will need to use to implement the processor.

There is a quirk in producing the PC (program counter) in which it will begin at 1 rather than 0. So this results in the first instruction in the memory array (which has the index of 0) will not be read.

To correctly implement the shift left and move function so it will accommodate the immediate input when needed, the result from the register for its A input will be held temporarily (tempa) as well as the B input. The controller will then check if the function is either move or shift left and that the operation requires the immediate input. If it requires immediate, the A input will be swapped with the B input so the immediate input will be processed instead of the read input.

2 Input/Output Port Description

Controller

Port Name	Port Size	Port Type	Description
clk	1	IN	Controls clock cycle

The following are each component implemented in the controller.

Register

Port Name	Port Size	Port Type	Description
clk	1	IN	Controls clock cycle (the same input in the controller)
din	5	IN	Accepts incremented PC from adder.
dout	5	OUT	Returns the PC on a positive clock edge.

Adder

Port Name	Port Size	Port Type	Description
in_0	5	IN	Accepts PC from register
output	5	OUT	Returns incremented PC for the register.

Memory

Port Name	Port Size	Port Type	Description
adr	6	IN	Accepts PC from register.
dout	32	OUT	Returns the 32-bit instruction set from the address the PC specified.

Register File

Port Name	Port Size	Port Type	Description
clk	1	IN	Controls clock cycle (the same input in the controller)
we	1	IN	Prevents data from being written in when the input function is invalid.
rs	6	IN	Accepts 6 bit address for the read source.
rt	6	IN	Accepts 6 bit address for the second read source.
rd	6	IN	Accepts 6 bit address for the write address.
opr1	32	OUT	Returns the 32 bit operand from the address designated by rs.

opr2	32	OUT	Returns the 32 bit operand from the address designated by rt.
wdata	32	IN	Accepts the 32 bit data to write in the address designated by rd.

Sign Extension

Port Name	Port Size	Port Type	Description
in_0	15	IN	Accepts 15 bit logic vector.
output	32	OUT	Returns the input with its sign extended to 32 bits.

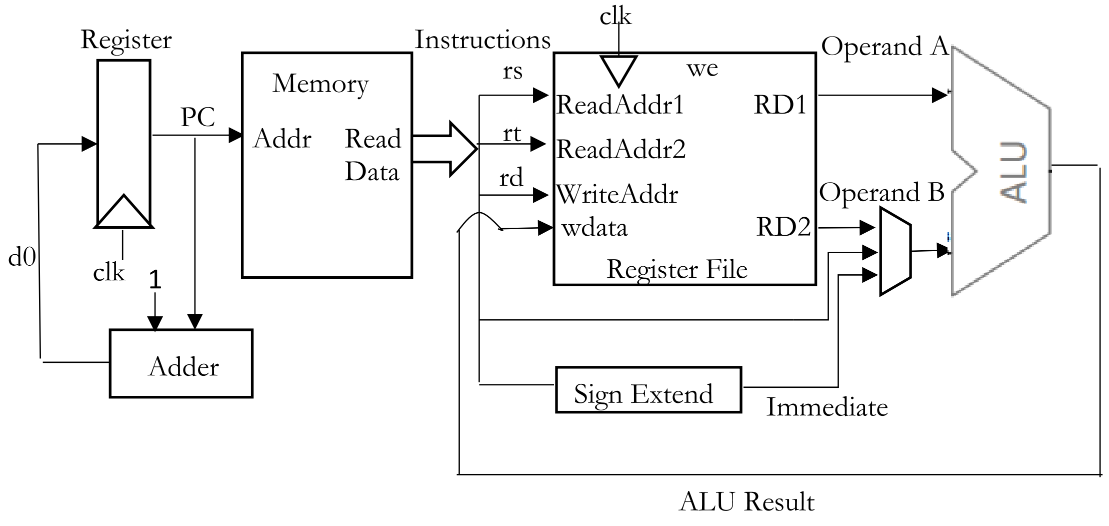
Operand B Multiplexor

Port Name	Port Size	Port Type	Description
r_i	1	IN	Determines if the instruction set is “read” or “immediate”
rt	32	IN	Accepts the 32-bit operand read by the register file.
imm	32	IN	Accepts the 32-bit immediate operand.
oprb	32	OUT	Returns operand according to r_i.

ALU 32-bit

Port Name	Port Size	Port Type	Description
A	32	IN	32 – bit input
B	32	IN	32 – bit input
opsel	3	IN	Determines operation on input A or between A and B
Mode	1	IN	Determines whether the operation will be from the arithmetic set or bit operator set
result	32	OUT	Operation output which will be written in the register file.
cout	1	OUT	Carry out bit if it applies (not used)

3 Design Schematic



Processor Design Implemented in the Controller

4 Expected and Simulation Waveform

Controller test bench running instructions that are pre-written.

For reference, here are all the instruction sets:

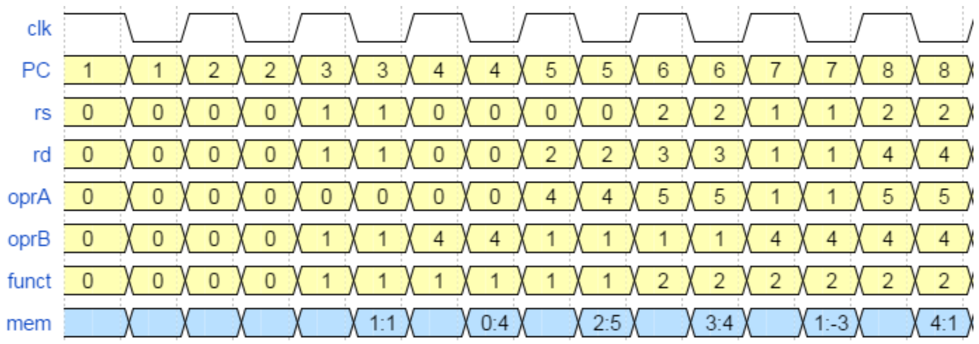
Read (0) Immediate(1)	Read Source Address	Write Address	Function	Immediate (15 bit) or Read Second Source (6 bit)
1	000001	000001	0001	000000000000001
1	000000	000000	0001	000000000000100
0	000000	000010	0001	000001
0	000010	000011	0010	000001
0	000001	000001	0010	000011
0	000010	000100	0010	000000
1	000010	000011	0101	000000000001001
1	000010	000101	0110	000000000001001
1	000101	000110	0111	UUUUUUUUUUUUUUU
1	000010	000111	1000	000000000001001
1	UUUUUU	001000	1001	101101110111101
1	UUUUUU	001001	1011	101101110111101
0	000101	111010	0101	000111
0	000011	111011	0110	000101
0	000110	111100	0111	UUUUUU

0	000011	111101	1000	111011
0	001000	111110	1001	UUUUUU
0	001001	111111	1011	UUUUUU

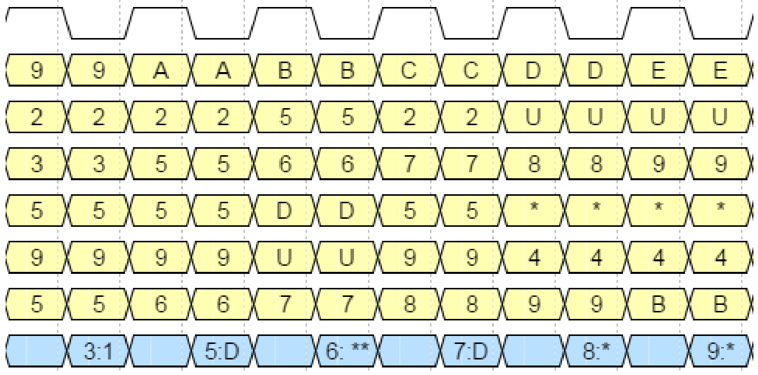
*Starting from PC = “00003”

Expected waveform is as follows:

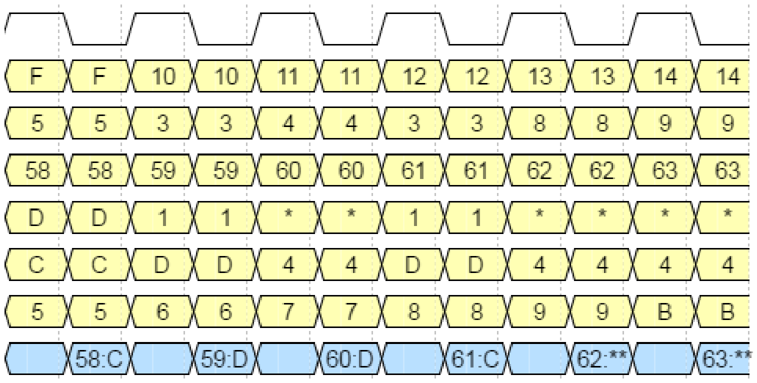
This covers processes involving addition and subtraction.



These processes cover instructions that uses the immediate operator.



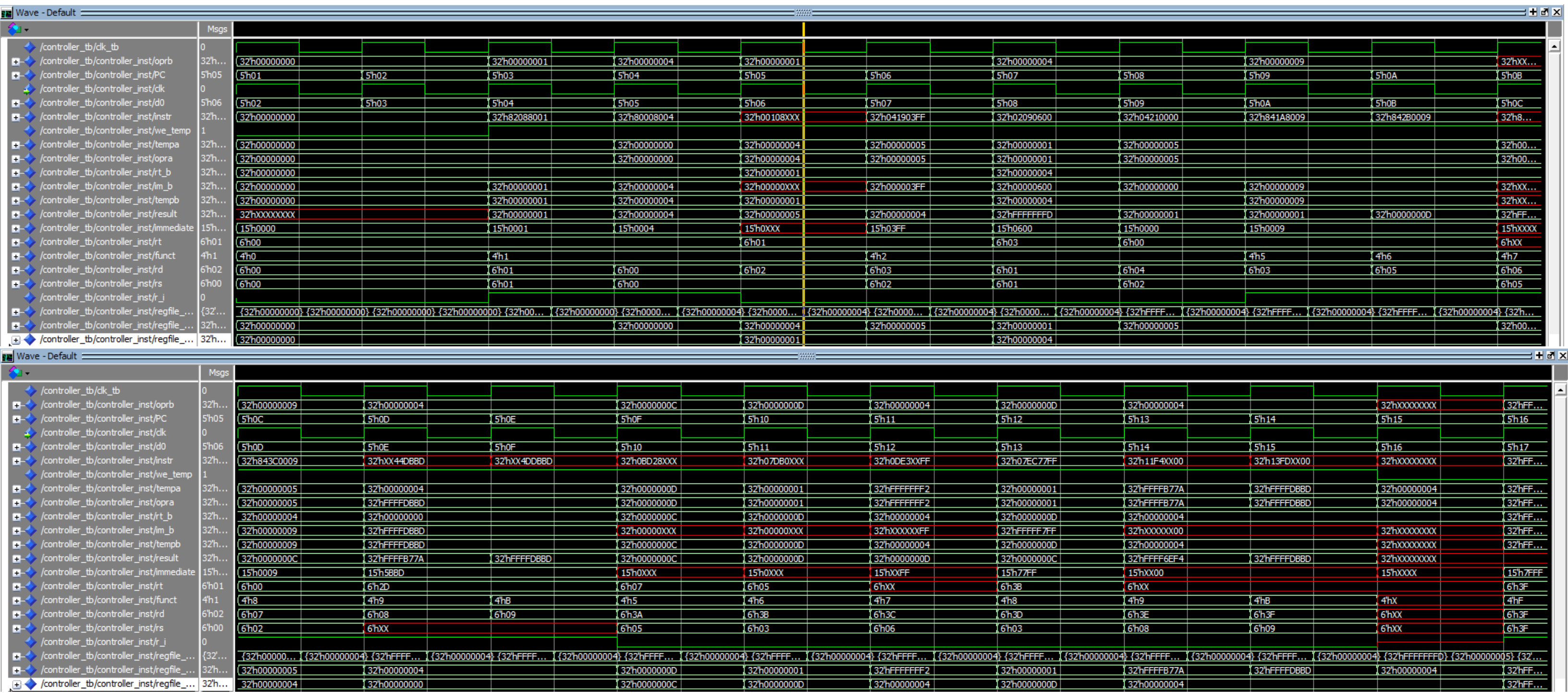
These processes cover instructions that uses the read operator.



Note: There are several important things to know about this waveform. First is that all the values are in hexadecimal. Second is that not all the signals and their values are listed on here. Third is that operand A will contain the immediate value only when the immediate instruction set is used with shift left and move operations. Fourth is that there are “U”s and straight “0”s and “1”s placed in certain places to demonstrate that their values ultimate does not change the result. Finally, it is important to know that in the memory, it is read like this: (memory index) : (value).

* First addressing oprA row and using PC as a reference to which column, at D, E, 13, and 14 the processor was using the shift left and move functions. D and E were given value “101101110111101” for its immediate value. At 13 and 14, the read values were the results of D and E which are stored in the memory. Therefore at 13 and 14, the values are “11111111111111111011011101111010” and “11111111111111111101101110111101” respectively (the extra ones are the result of the sign extension because it’s an immediate value) which are also the values kept in the memory index D and E. At 13 and 14 in memory indexes 62 and 63 are “111111111111111110110111011110100” and “11111111111111111101101110111101” respectively. At B, the complement is taken from D, resulting in $\overline{D}$. Likewise in 11, oprA will be $\overline{D}$ resulting in D written in the memory.

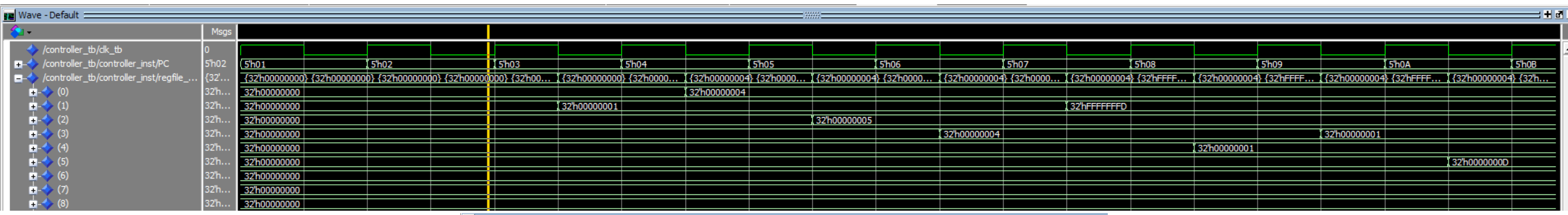
Simulation waveform is as follows:

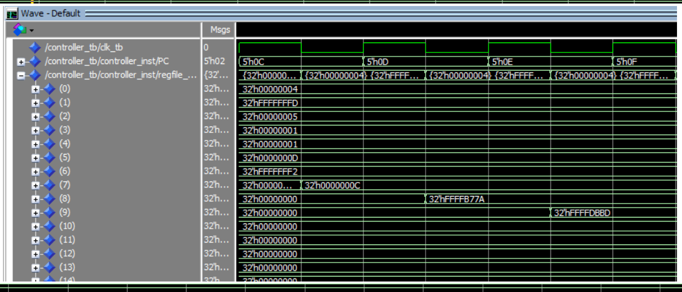


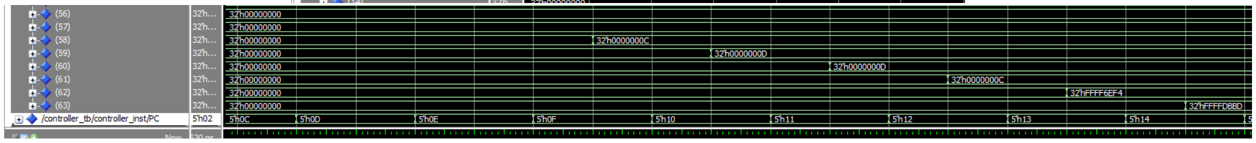
Wave - Default
Msgs
/controller_tb/clk_tb
/controller_tb/controller_inst/oprb
/controller_tb/controller_inst/PC
/controller_tb/controller_inst/clk
/controller_tb/controller_inst/d0
/controller_tb/controller_inst/instr
/controller_tb/controller_inst/we_temp
/controller_tb/controller_inst/tempa
/controller_tb/controller_inst/opra
/controller_tb/controller_inst/rt_b
/controller_tb/controller_inst/im_b
/controller_tb/controller_inst/tempb
/controller_tb/controller_inst/result
/controller_tb/controller_inst/immediate
/controller_tb/controller_inst/rt
/controller_tb/controller_inst/funct
/controller_tb/controller_inst/rd
/controller_tb/controller_inst/rs
/controller_tb/controller_inst/r_i
/controller_tb/controller_inst/regfile_...

Note that everything after the PC is 14 that the instruction sets are not valid, so there are things that are not working as intended.

The memory as well as the PC is as follows:


Wave - Default
Msgs
/controller_tb/clk_tb
/controller_tb/controller_inst/PC
/controller_tb/controller_inst/regfile_...
(0)
(1)
(2)
(3)
(4)
(5)
(6)
(7)
(8)
5'h01
5'h02
5'h03
5'h04
5'h05
5'h06
5'h07
5'h08
5'h09
5'h0A
5'h0B
32'h00000000
32'h00000004
32'h00000001
32'hFFFFFFFD
32'h00000005
32'h00000004
32'h00000001
32'h00000001
32'h0000000D


Wave - Default
Msgs
/controller_tb/clk_tb
/controller_tb/controller_inst/PC
/controller_tb/controller_inst/regfile_...
(0)
(1)
(2)
(3)
(4)
(5)
(6)
(7)
(8)
(9)
(10)
(11)
(12)
(13)
5'h0C
5'h0D
5'h0E
5'h0F
32'h00000004
32'hFFFFFFFD
32'h00000005
32'h00000001
32'h00000001
32'h0000000D
32'hFFFFFFF2
32'h0000000C
32'hFFFFB77A
32'hFFFFDBBD
32'h00000000


(56)
(57)
(58)
(59)
(60)
(61)
(62)
(63)
/controller_tb/controller_inst/PC
5'h02
32'h00000000
32'h0000000C
32'h0000000D
32'h0000000D
32'h0000000C
32'hFFFF6EF4
32'hFFFFDBBD
5'h0C
5'h0D
5'h0E
5'h0F
5'h10
5'h11
5'h12
5'h13
5'h14