Lab 5: Building a Simple Processor Part 5

OVERVIEW

The processor we have designed for lab 5 is extended to support the following instructions.

- mult – signed multiplication of 2 values in registers and place the result in another register.
- sll – apply logical shift to left (0 bits are pushed) to a value in a register and place the result in another register.
- srl - apply logical shift to right (sign bit is not extended) to a value in a register and place the result in another register.
- sra - Apply arithmetic shift to right (sign bit is extended) to a value in a register and place the result in another register.
- ror - Rotate a value placed in a register.
- bne- branch to another instruction if two register values are not equal.

ALU opcodes

Opcode	ALU function
000	Forward
001	Add
010	And
011	Or
100	multiplication
101	Logical shift left
110	Arithmetic shift right
111	Rotate right

Instruction Encodings, Assigned Opcodes, and Control Signals

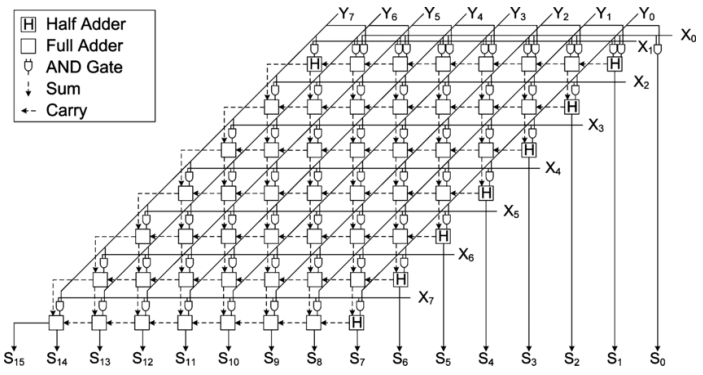
- Reverse control signal was introduced to support sll and srl instructions since they function on the same functional unit.
- Since branch instructions (bne and beq) use the same functional units, branch opcode uses a 2-bit signal for the selection.
    - 00 – Normal flow
    - 01 - beq
    - 10 – bne

Instruction	Instruction Opcode	Control Signals						
		Write enable	Comp select	Immediate select	ALU opcode	Jump	Reverse	Branch
loadi	0000_0000	1	0	1	000	0	0	00
mov	0000_0001	1	0	0	000	0	0	00
add	0000_0010	1	0	0	001	0	0	00
sub	0000_0011	1	1	0	001	0	0	00
and	0000_0100	1	0	0	010	0	0	00
or	0000_0101	1	0	0	011	0	0	00
j	0000_0110	0	0	0	000	1	0	00
beq	0000_0111	0	1	0	001	0	0	01
mult	0000_1100	1	0	0	100	0	0	00
sll	0000_1101	1	0	1	101	0	0	00
srl	0000_1110	1	0	1	101	0	1	00
sra	0000_1111	1	0	1	110	0	0	00
ror	0001_0000	1	0	1	111	0	0	00
bne	0001_0001	0	1	0	001	0	0	10

New Instructions

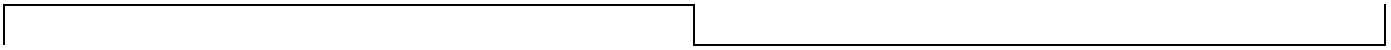
Signed multiplication -mult instruction

Since multiplication is done for sign numbers and the MSB is the sign bit, we use six 7-bit adders to get the multiplication value. And using the sign bits of two numbers we can get the sign of multiplication.



In the multiplication module, the value of the multiplication is basically built using gate-level modeling. Thus, 7-bit adders are implemented using full adders and half adders. Using the MSB of each data line the input of the adder will vary two's complement or not. Thus, we can multiply both sign and unsigned numbers.

Timing Diagram

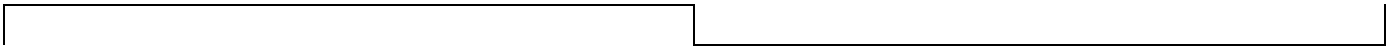


PC Update	Instruction Memory Read		Register read	ALU	
#1	#2		#2	#2	
	PC + 4 Add		Branch/jump Target		
	#1		#2		
Register write			Decode		
#1			#1		

Logical Shift Left and logical shift right – sll and srl instructions

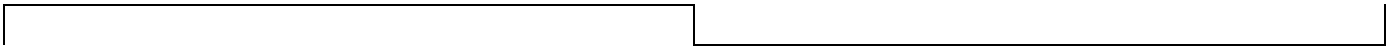
The logical shift module performs a logical shift left by a given number of times. To perform a shift right another module is introduced to reverse a given number and the reverse control bit controls the reverse module. When performing shift right operation, first the number is reversed by the reverse module and then input to the logical shift sub-module in ALU. By reversing the result from the ALU, the final result with the right-shifted value is obtained.

Timing Diagram – sll



PC Update	Instruction Memory Read		Register read	ALU	
#1	#2		#2	#2	
	PC + 4 add		Branch/jump Target		
	#1		#2		
Register write			Decode		
#1			#1		

Timing diagram – srl

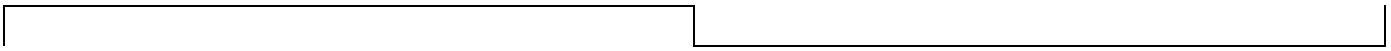


PC Update	Instruction Memory Read		Register read	Reverse	ALU	Reverse
#1	#2		#2	#1	#1	#1
	PC + 4 Add		Branch/jump Target			
	#1		#2			
Register write			Decode			
#1			#1			

Arithmetic Shift right – sra instruction

Arithmetic shift right is performed by preserving MSB which is the sign bit. To perform this, a similar module to the logical shift module is used but the right bits are filled with MSB of the 8-bit value.

Timing Diagram

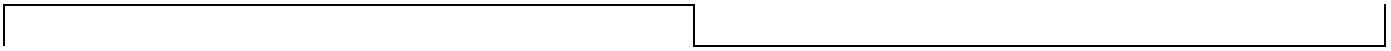


PC Update	Instruction Memory Read		Register read	ALU	
#1	#2		#2	#1	
	PC + 4 Add		Branch/jump Target		
	#1		#2		
Register write			Decode		
#1			#1		

Rotate right – ror instruction

The input is rewired to the result by the given amount or rotation.

Timing diagram



PC Update	Instruction Memory Read		Register read	ALU	
#1	#2		#2	#1	
	PC + 4 Add		Branch/jump Target		
	#1		#2		
Register write			Decode		
#1			#1		

Branch not equal – bne instruction

Similar to the beq instruction but PC is updated with the target address if zero output of ALU is 0. 2-bit control signal is given to MUX which selects the next PC address to distinguish normal flow (PC+4) or bne, beq instructions (PC + 4 + target offset)

Timing Diagram

PC Update	Instruction Memory Read		Register read	2's Comp	ALU
#1	#2		#2	#1	#2
	PC + 4 Add		Branch/jump Target		
	#1		#2		
Register write			Decode		
#1			#1		

Datapath and Control flow Diagram