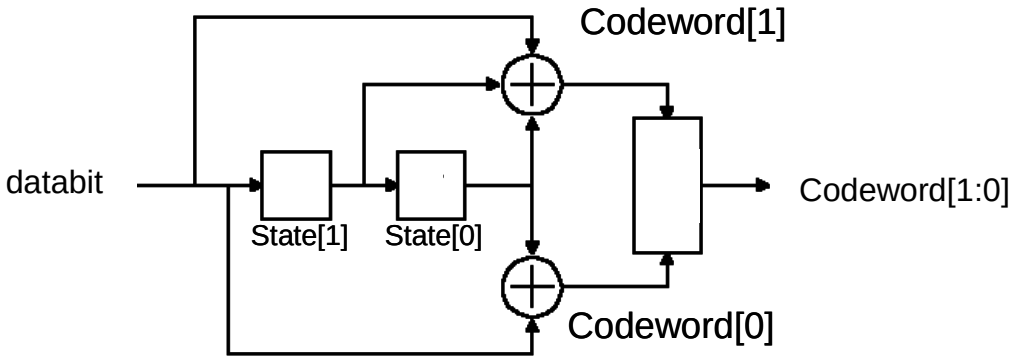
Name: _____ **Student #:** _____ **ID: A**

CSE 2021 - Quiz 3 - Fall 2009

Problem

1. Convolutional encoding is commonly used on cell phones to ensure data is received with few errors using the low transmit power typically available on a cell phone. A convolutional encoder is to be included in the core of a cell phone processor and requires a Verilog module to be written.

   The module will use a finite state machine approach and will accept two inputs - databit and state[1:0] and produce two outputs codeword[1:0] and nextstate[1:0]. The encoding process is diagrammed below, which shows the databit and state bits. The process outputs 2 bits for every 1 input bit. The nextstate[1:0] is defined by [databit : state 1]. In other words, the bits are shifted to the right to continue the process. The codeword bits are formed by XOR'ing the indicated bits together.



   (a) What type of finite state machine is this and why?

   (b) Fill out the following table for nextstate and codeword outputs

Name: _____

databit	state[1:0]	nextstate[1:0]	codeword[1:0]
0	00		
0	01		
0	10		
0	11		
1	00		
1	01		
1	10		
1	11		

(c) Write the conv module based on the above description.

```
module conv(state, databit, nextstate, codeword);




endmodule
```

(d) A test bench Verilog module has been written (see below). What is the output from this?

```
module testbench;
```

Name: _____

```
reg [1:0] state;
reg [3:0] test_data;
reg test_bit;
wire [1:0] codeword, nextstate;
integer i;

conv uut_conv(state, test_bit, nextstate, codeword);

initial
begin

 test_data = 4'b1001;
 state = 2'b00;

 for (i = 0; i < 4; i = i+1)
  begin
   #200
   test_bit = test_data[i];
   #1 $display($time, " databit = %b, state = %b, nextstate = %b codeword = %b",test_bit,
state, nextstate, codeword);

   state = nextstate;
   end
  $finish;
  end
endmodule
```

Output:

Name: _____

2. (a) Enter an appropriate value in each table cell corresponding to the Instruction in the column. Enter an 'X' if the value is not applicable for the instruction. Note there is a format for the control bits (see note below the table) and there is reference figures and tables to help you in the following pages.

ID		*EX*		*MEM*		*WB*	
Ins	`lw $3, 0($sp)`	*Ins*	`or $7, $2, $4`	*Ins.*	`sub $3, $20, $3`	*Ins.*	`lw $20 0($1)`
WB[1]		*WB*[1]		*WB*[1]		*WB*[1]	
Mem[1]		*Mem*[1]		*MEM*[1]		*Write Reg*	
EX[1]		*EX*[1]		*Address*		*Write Data*[2]	
rs		*AddResult*		*Write Data*			
rt		*ALUResult*		*WB Reg*			
rd		*Zero*					
operand		*rt*					
		rd					

Notes:

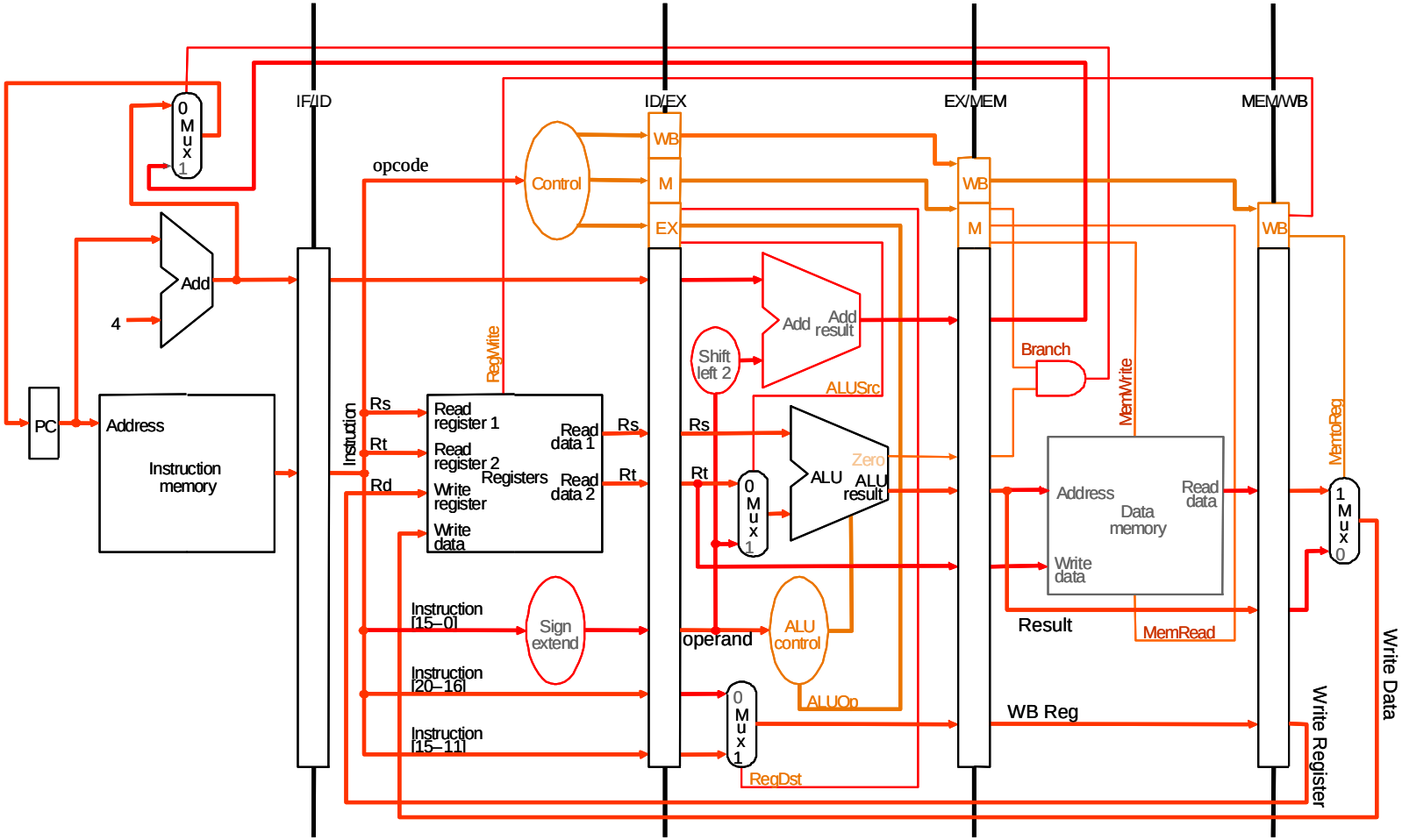
1. Format for control bits: **EX**: RegDst, ALUOp[1:0], ALUSrc, **Mem**: Branch, MemRead, MemWrite, **WB**: MemtoReg
2. Provide the expression for the contents of this portion of the pipeline register. The format of the expression should be the same as given on the MIPS Reference Data sheet, "Core Instruction Set" table, "Operation (in Verilog)" column for the instruction in the question.

Name: _____

(b) Identify any data hazards by filling out the following table (an example is shown for illustration only):

Hazards				
Instruction 1	*Instruction 2*	*Ins 1 REG.item*	*Ins 2 REG.item*	*Stall or Forward?*
`add $3, $2, $1`	`sub $5, $4, $3`	*EX/MEM.Rd*	*FD/EX.Rt*	*Forward*

Name: _____



Reference Figure - 1

Name: _____

Reference Table - 1

Instruction opcode	ALUOp	Instruction operation	Function code	Desired ALU action	ALU control input
LW	00	load word	XXXXXX	add	0010
SW	00	store word	XXXXXX	add	0010
Branch equal	01	branch equal	XXXXXX	subtract	0110
R-type	10	add	100000	add	0010
R-type	10	subtract	100010	subtract	0110
R-type	10	AND	100100	AND	0000
R-type	10	OR	100101	OR	0001
R-type	10	set on less than	101010	set on less than	0111

Reference Table - 2

Instruction	Execution/address calculation stage control lines				Memory access stage control lines			Write-back stage control lines	
	RegDst	ALUOp1	ALUOp0	ALUSrc	Branch	Mem-Read	Mem-Write	Reg-Write	Memto-Reg
R-format	1	1	0	0	0	0	0	1	0
lw	0	0	0	1	0	1	0	1	1
sw	X	0	0	1	0	0	1	0	X
beq	X	0	1	0	1	0	0	0	X

CSE 2021 - Quiz 3 - Fall 2009
Answer Section

PROBLEM

1. ANS:
(a) Meely machine - next state depends on both the input and the current state.

(b)

databit	state[1:0]	nextstate[1:0]	codeword[1:0]
0	00	00	00
0	01	00	11
0	10	01	10
0	11	01	01
1	00	10	11
1	01	10	00
1	10	11	01
1	11	11	10

(c)

```
module conv(state, databit, nextstate, codeword);
     input [1:0] state;
     input databit;
     output[1:0] nextstate, codeword;


     assign nextstate[1] = databit;
     assign nextstate[0] = state[1];

     assign codeword[1] = databit^state[1]^state[0];
     assign codeword[0] = databit^state[0];

endmodule
```

(d)

```
C:\Verilog>
C:\Verilog>iverilog testb-conv.v conv.v

C:\Verilog>vvp a.out
                 201 databit = 1, state = 00, nextstate = 10 codeword = 11
                 402 databit = 0, state = 10, nextstate = 01 codeword = 10
                 603 databit = 0, state = 01, nextstate = 00 codeword = 11
                 804 databit = 1, state = 00, nextstate = 10 codeword = 11
```

PTS: 1

2. ANS:
(a)

ID		*EX*		*MEM*		*WB*	
Ins	`lw $3, 0($sp)`	*Ins*	`or $7, $2, $4`	*Ins.*	`sub $3, $20, $3`	*Ins.*	`lw $20 0($1)`
WB[1]	*0*	*WB*[1]	*0*	*WB*[1]	*0*	*WB*[1]	*1*
Mem[1]	*X 1 0*	*Mem*[1]	*X 0 0*	*MEM*[1]	*X 0 0*	*Write Reg*	*20*
EX[1]	*0 00 1*	*EX*[1]	*1 10 0*	*Address*	*X*	*Write Data*[2]	*M[R[rs]+SignExtImm] or M[R[$1]+0]*
rs	*29 (or $sp or R[$29])*	*AddResult*	*X*	*Write Data*	*X*		
rt	*3 or R[$3]*	*ALUResult*	*R[rs] | R[rt] or R[$2]|R[$4]*	*WB Reg*	*3*		
rd	*X*	*Zero*	*X*				
operand	*0*	*rt*	*X (or R[$4] or $4*				
		rd	*7*				

(b)

Hazards				
Instruction 1	*Instruction 2*	*Ins 1 REG.item*	*Ins 2 REG.item*	*Stall or Forward?*
`add $3, $2, $1`	`sub $5, $4, $3`	*EX/MEM.Rd*	*FD/EX.Rt*	*Forward*
`lw $20 0($1)`	`sub $3, $20, $3`	*ID/EX.Rd*	*IF/ID.Rs*	*Stall*

PTS: 1