COMP 22111

Two hours

UNIVERSITY OF MANCHESTER
SCHOOL OF COMPUTER SCIENCE

Processor Microarchitecture

Date: Wednesday 18th January 2017

Time: 14:00 - 16:00

Please answer any THREE Questions from the FOUR Questions provided

Use a SEPARATE answerbook for EACH Question.

This is a CLOSED book examination

The use of electronic calculators is permitted provided they are not programmable and do not store text

[PTO]

1.

a) Figure 1 lists the Verilog code for a Stump function, Testbranch, which is similar to what you have seen in the laboratory.

i) What is the difference between a Verilog `task` and a Verilog `function`? (1 mark)

ii) In what circumstances would you use either a `task` or a `function`? (1 mark)

iii) Where would you locate a `task` or `function` in your Verilog code? (1 mark)

iv) What operation is being performed using `~(C|Z)` on line 13? (1 mark)

v) During simulation of Testbranch the variable `condition` is set to `X`, what will be the result of this with respect to the value returned by Testbranch? How could you modify the code to produce a more desirable response? (2 marks)

b) Sketch a block diagram to illustrate the elements of a Verilog test bench, where the test bench is designed to indicate an error when the output differs from golden test data. (4 marks)

c) Figure 2 shows the design for a four bit shift register with parallel load and parallel readout that has been instantiated as a Verilog module using structural Verilog. The input `sel` controls the operation of the shift register. When `sel` = 0 the shift register is loaded using the parallel data on `D` on a rising edge of `clk`. When `sel` = 1 the contents of the shift register are shifted by one bit to the right on the rising edge of `clk`.

The shift register design has been implemented as a Verilog module with module definition

```
module shift_4bit (input  [3:0]   D,
                   output [3:0]   Q,
                   input          sel,
                   input          clk,
                   input          rst);
```

(Question 1 continues on the following page)

(Question 1 continues from the previous page)

You are required to produce a Verilog testbench to test the operation of the shift register design. Figure 3 shows the definition of the test module. Produce the missing code as illustrated. You may assume that the components within the design of the shift register have been tested extensively and operate to the required specification.

(10 marks)

```
1:   function Testbranch;
2:
3:   input [3:0] condition;
4:   input [3:0] CC;
5:
6:   reg N, Z, V, C;
7:
8:   begin
9:    {N,Z,V,C} = CC;
10:   case (condition)
11:    0 :  Testbranch =   1;
12:    1 :  Testbranch =   0;
13:    2 :  Testbranch = ~(C|Z);
14:    3 :  Testbranch =   C|Z;
15:    4 :  Testbranch =  ~C;
16:    5 :  Testbranch =   C;
17:    6 :  Testbranch =  ~Z;
18:    7 :  Testbranch =   Z;
19:    8 :  Testbranch =  ~V;
20:    9 :  Testbranch =   V;
21:    10 : Testbranch =  ~N;
22:    11 : Testbranch =   N;
23:    12 : Testbranch =   V^~N;
24:    13 : Testbranch =   V^N;
25:    14 : Testbranch = ~((V^N)|Z) ;
26:    15 : Testbranch =  ((V^N)|Z) ;
27:   endcase
28:  end
29:
30:  endfunction
```

Figure 1

(Question 1 continues on the following page)

(Question 1 continues from the previous page)

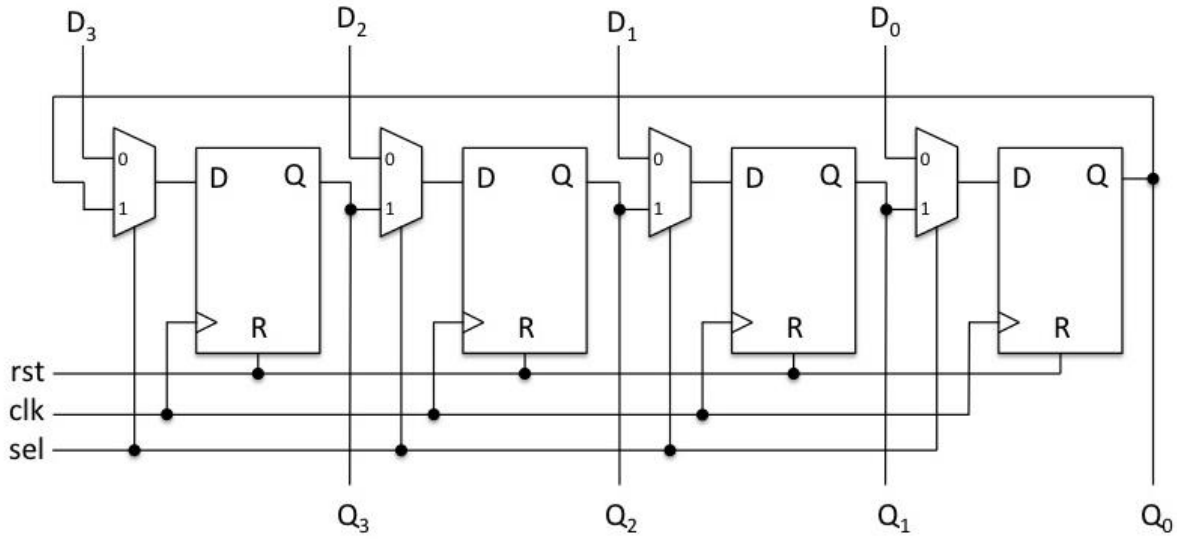

Figure 2

```
module shift_reg_test ();

reg        op_sel; // operation for DUT
reg        clock;  // clock signal to DUT
reg        reset;  // reset to DUT
reg  [3:0] D_data; // parallel input to DUT
wire [3:0] Q_data; // parallel output from DUT




////////////////////////////////////////
// write the missing code that goes here
////////////////////////////////////////




endmodule
```

Figure 3

2.

a) The Stump register bank has a register that is hard-wired to zero. Why is this design feature adopted? Give two examples illustrating the use of this register. (2 marks)

b) Stump is a RISC processor that adopts a load/store architecture. Explain what a load/store architecture means. What is included within the Stump instruction set to support such an architecture? (2 marks)

c) Stump has two types of instruction that differ depending on the source of the data operands. Discuss where operands come from for these two instruction types. How does the control block determine which type of instruction is being executed? (3 marks)

d) Figure 4 lists some Stump assembly code for performing a widely used function. All values are given in hexadecimal and initially the PC is 0021_{16}.

i) Discuss what the instructions at memory addresses 0x0021 to 0x0023 are doing. (1 mark)

ii) Memory address 0x0026 contains a branch instruction (shown as `Bxx`). What branch instruction is required to perform the function? What function does the code implement? (2 marks)

iii) For the branch instruction at memory address 0x0026, what offset would be required to give the correct branch address? (1 mark)

e) Figure 5 shows an RTL datapath design for the Stump processor, where control signals to the functional components are indicated. Tables 1, 2 and 3 list the possible control signals for alu_func, shift_op and sign_ext_op. Produce "signal usage charts" that tabulate the status of all the control signals (using 'X' where appropriate) for the following instructions from Figure 4:

i) the **fetch** state for the instruction at memory address 0x0025, (3 marks)

ii) the **execute** state for the instruction at memory address 0x0028, (4 marks)

iii) the **memory** state for the instruction at memory address 0x0027. (2 marks)

An example signal usage chart is given in Table 4.

(Question 2 continues on the following page)

(Question 2 continues from the previous page)

```
ORG 0x0021
     ADD  R2, R0, R0
     LD   R4, [R0, #30]
     LD   R5, [R0, #31]
loop ADD  R2, R2, R4
     SUBS R5, R5, #1
     Bxx  loop
     ST   R2, [R0, #32]
stop BAL  stop

ORG 0x0030
     Data 0x004
     Data 0x003
     Data 0x000
```

Figure 4

ALU Operation	alu_func
ADD	000
ADC	001
SUB	010
SBC	011
AND	100
OR	101
LD/ST	110
Bcc	111

Table 1

Shift operation	shift_op
No Shift	00
ASR	01
ROR	10
RRC	11

Table 2

Sign Extender Operation	sign_ext_op
5 – 16 bit	0
8 – 16 bit	1

(Question 2 continues on the following page)

(Question 2 continues from the previous page)

Table 3

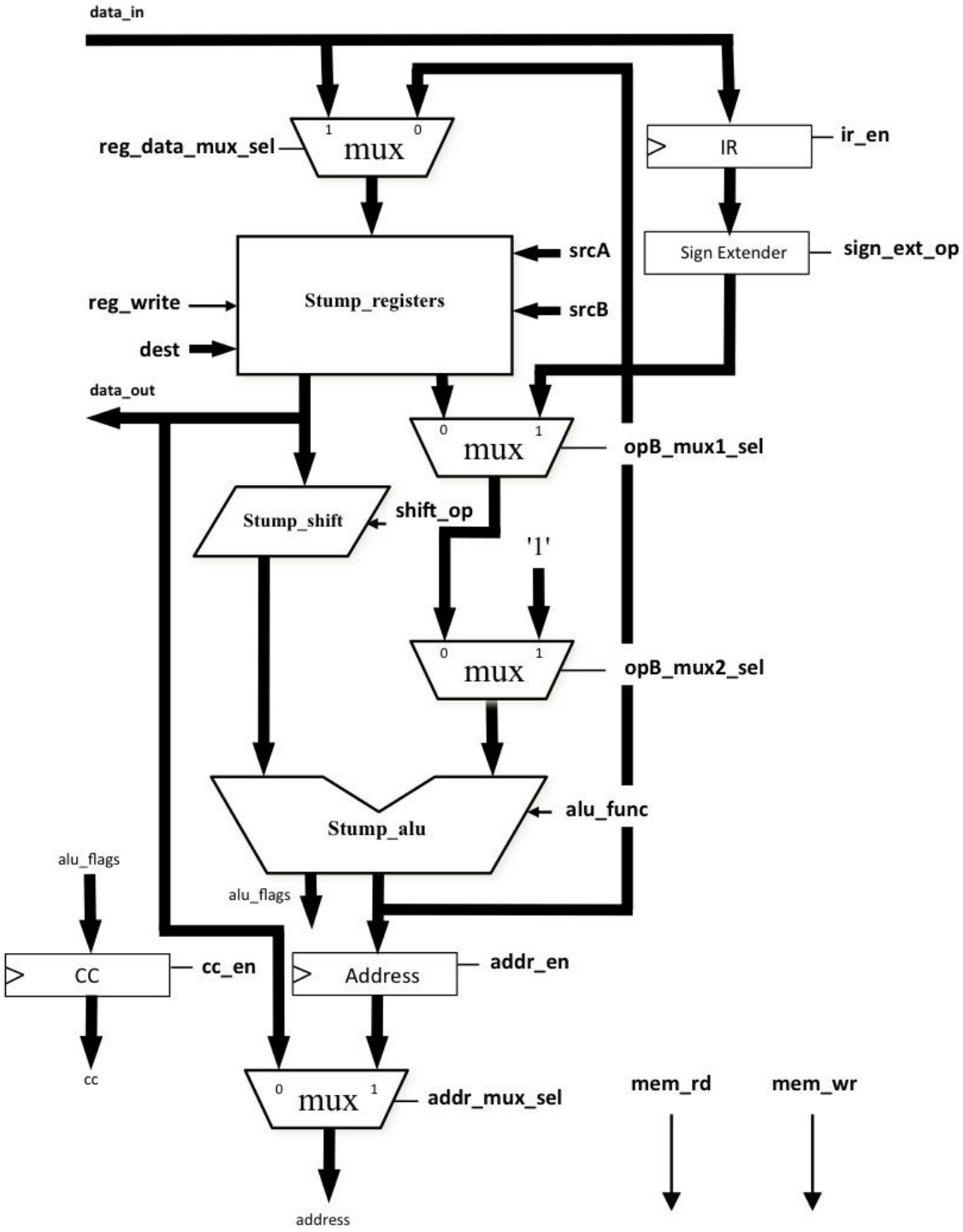

Figure 5

(Question 2 continues on the following page)

(Question 2 continues from the previous page)

Control signal	Status
ir_en	
reg_data_mux_sel	
sign_ext_op	
srcA	
srcB	
reg_write	
dest	
shift_op	
opB_mux1_sel	
opB_mux2_sel	
alu_func	
cc_en	
addr_en	
addr_mux_sel	
mem_rd	
mem_wr	

Table 4

3.

a) On completion of an arithmetic addition, on a typical processor (e.g. Stump, ARM, x86 etc.) what (if anything) may be indicated by the state of:

i) the carry flag? (1 mark)
ii) the overflow flag? (1 mark)

b) What is meant by "saturating" arithmetic? Give a numeric example which exhibits the difference between a saturating and non-saturating operation. (You may choose your own radix, word-length, operation etc. but make sure your choices are clearly stated in the answer.) (2 marks)

Operand sizes vary according to the application. Sometimes operands may be longer than the processor's word length; in other applications, particularly 'multimedia', short integers such as 16- or even 8-bit values are adequate. In the latter case operations are frequently repeated many times on 'vectors' of short values. Unlike Stump, a typical modern CPU will have a 32- or 64-bit wide datapath: i.e. its registers, ALU(s) etc. all handle 32 or 64 bit words in parallel.

c) Write some (commented) Stump assembly language code to add two 32-bit signed numbers. You may assume the operands are already in processor registers and that the result can also remain in registers. (Indicate what the registers are used for in your answer.) (4 marks)

d) As well as *arithmetic* operation status, what else is a typical processor (including Stump, ARM, x86) carry flag used for? (2 marks)

e) A common way of extending a processor instruction set is to add some 'SIMD' instructions. Explain what this term means. How is it applied when enhancing a CPU for 'multimedia' applications? (4 marks)

f) To facilitate a SIMD 'ADD' operation, what changes (if any) from the simple ("scalar") operation might need to be applied to the:

i) register bank? (2 marks)
ii) ALU? (2 marks)

g) How have processor manufacturers extended the SIMD concept to give potentially still greater acceleration of 'multimedia' code? (2 marks)

[PTO]

4.

The Stump architecture is quite primitive and lacks some types of instructions that are common on commercial ISAs. One example is the means of invoking a subroutine (method).

- In the ARM ISA this is done with a "BL" instruction which branches to the specified code address but leaves the address following the call in a reserved ('link') register.

- In the x86 ISA this is done with a "CALL" instruction which branches to the specified code address but pushes the address following the call onto the stack. (A stack push involves a store operation and a modification to the stack pointer.)

a) Contrast the relative advantages of each of these mechanisms. (4 marks)

b) You are asked to add an instruction similar to 'BL' to the stump ISA. Using the datapath microarchitecture as given in Figure 5 describe a sequence of internal operations which could perform this. (6 marks)

c) How could the Stump 'return' from the called routine? (Write the mnemonic(s) if that helps.) (2 marks)

d) Existing Stump branches are conditional and use short, PC-relative signed offsets. (Such instructions are quite common in other ISAs - e.g. x86 has a similar class of operations.) Briefly discuss the suitability of these characteristics for BL instructions. (4 marks)

e) Adding a Stump BL instruction also needs an instruction encoding. The existing coding for branch instructions and the unused instruction code space are shown in Table 5. Suggest a reasonable *encoding* for BL operations bearing in mind your answer to the previous section. Where appropriate, justify the reasons for your choices. (4 marks)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Branch	1	1	1	1	condition				offset							
Reserved	1	1	1	0	unused											

Table 5

END OF EXAMINATION