The University of Alabama in Huntsville
Electrical and Computer Engineering Department
CPE 221 01
Test 2 Solution
Fall 2019

1. (2 points) List the RTL for the instruction POP (r0) _**r0 ← M[sp], sp ← sp + 4**_

2. (1 point) The stack pointer always points to the _**top**_ of the stack.

3. (1 point) _**Pipelining**_ is a technique in which the execution of multiple instructions are overlapped to increase the number of instructions executed in a period of time.

4. (1 point) True/False _**True**_ The stack pointer can change during the execution of a procedure.

5. (8 points) A processor executes an instruction in the following five stages. The time required by each stage in picoseconds (1,000 ps = 1 ns) is:

F	Fetch	170 ps
O	Decode/Read Operands	120 ps
E	Execute	250 ps
M	Memory	380 ps
WB	Result Writeback to Register	120 ps

   a. (4 points) What is the time taken to fully execute an instruction assuming that this structure is pipelined in five stages and that there is an additional 10 ps per stage due to the pipeline latches?
   $T_{pipelined}$ = 5*(Max(F, O, E, M, WB) + Pipeline Latch) = 5*(MAX(170 ps, 120 ps, 250 ps, 380 ps, 120 ps) + 10 ps) = 5*(380 ps + 10 ps) = 5* 390 ps = 1950 ps = 1.95 ns

   b. (4 points) What is the time to execute an instruction if the processor is not pipelined?
   $T_{non\text{-}pipelined}$ = F + O + E + M + WB = (170 + 120 + 250 + 380 + 120) ps = 1040 ps = 1.04 ns.

6. (20 points) Write the code to implement the expression A = (B – C)/(D+(E+F)/G) on 3-, 2-, 1-, and 0-address machines. Do not rearrange the expression. In accordance with programming language practice, computing the expression should not change the values of its operands. When using a 0-address machine, the order used is SOS op TOS, where SOS is second on stack and TOS is top of stack.

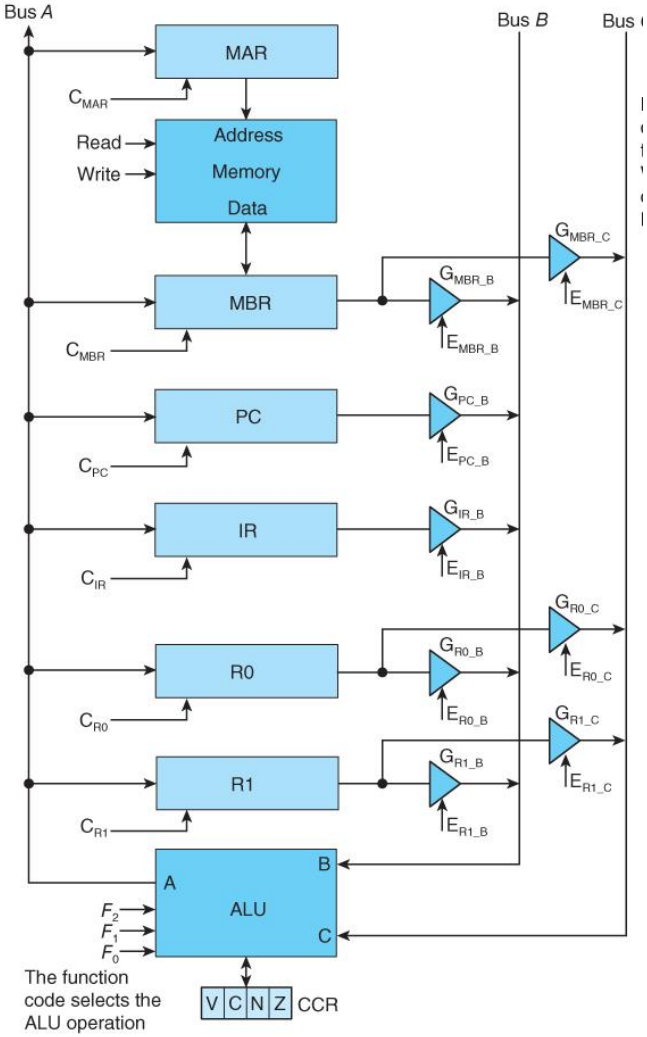
3-address	2-address	1-address	0-address

SUB A, B, C
ADD T, E, F
DIV T, T, G
ADD T, T, D
DIV A, A, T
	LOAD A, B
	SUB A, C
	LOAD T, E
	ADD T, F
	DIV T, G
	ADD T, D
	DIV A, T
		LDA E
		ADD F
		DIV G
		ADD D
		STA A
		LDA B
		SUB C
		DIV A
		STA A
			PUSH B
			PUSH C
			SUB
			PUSH E
			PUSH F
			ADD
			PUSH G
			DIV
			PUSH D
			ADD
			DIV
			POP A

7. (30 points) Complete the ARM assembly language program below so that it implements the following C++ statements. Create a subroutine cube that expects the input parameter to be in register r8 and returns the result in register r9. There is no need to use the stack.

```
;
;       This program calculates the cube of each of the elements of an
;       array using a subroutine and stores them in another array.
;       int cube(int);
;       void main(void)
;       {
;         const int size = 10;
;         int x[size] = {100, 3, -1, 2, 4, 4, 2, -1, 3, 100};
;         int z[size];
;         int i;
;         for (i = 0; i < size; i++)
;         {
;           z[i] = cube(x[i]);
;         }
;       int cube(int val)
;       {
;         int i, result;
;         result = 1
;         for (i = 0; i < 3; i++)
;           result = result * val;
;         return result;
;       }
        AREA    PROB_8, CODE, READ
        ENTRY
        ADR     r0, x
        ADR     r2, z
        LDR     r3, size
        LDR     r4, i
loopa   CMP     r4, r3
        BGE     done
        LDR     r8, [r0, r4, LSL #2]
        BL      cube
        STR     r9, [r2, r4, LSL #2]
        ADD     r4, r4, #1
        B       loopa
done    B       done
cube    MOV     r9, #1
        MOV     r1, #0
loopb   CMP     r1, #3
        BGE     back
        MUL     r9, r8, r9
        ADD     r1, r1, #1
        B       loopb
back    MOV     pc, lr
done    B       done
x       DCD     100, 3, -1, 2, 4, 4, 2, -1, 3, 100
z       SPACE   40
i       DCD     0
size    DCD     10
        END
```

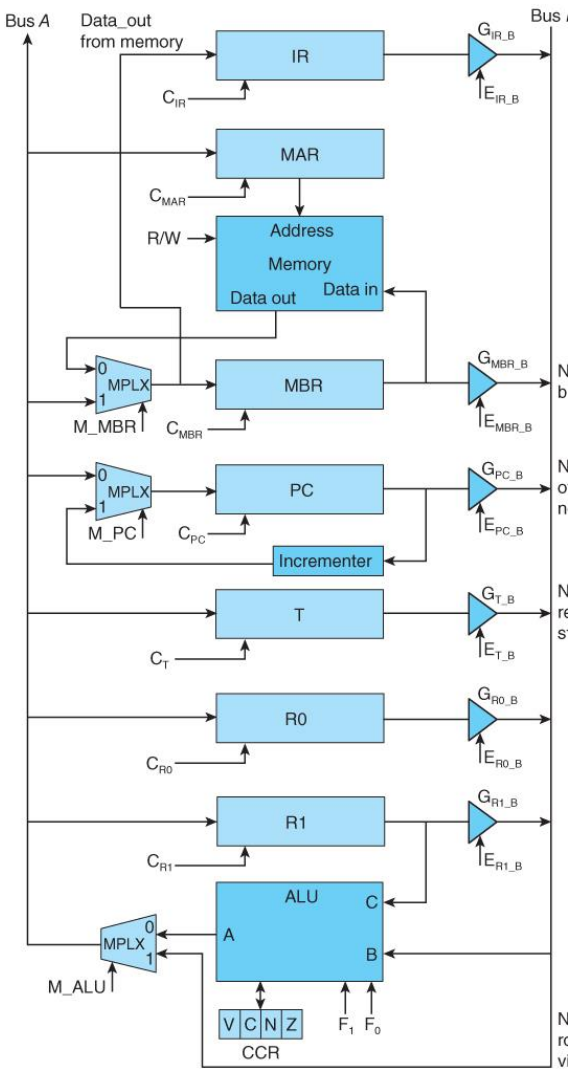
8. (15 points) For the architecture shown, write the sequence of signals and control actions necessary to execute the instruction `SUB R1, R1, R0`, that stores R1 – R0 in R1 The actions of this instruction are described by the abstract RTL `R1 ← R1 - R0`.



F_2	F_1	F_0	Operation
0	0	0	A = B'
0	0	1	A = C
0	1	0	A = B + 1
0	1	1	A = B + C
1	0	0	A = C
1	0	1	A = C
1	1	0	A = C
1	1	1	A = C

Cycle	Concrete RTL	Signals
1	**MBR ← R0'**	**C_{MBR}, E_{R0_B}, F = 000**
2	**MBR ← MBR + 1**	**C_{MBR}, E_{MBR_B}, F = 010**
3	**R1 ← R1 + MBR**	**C_{R1}, E_{MBR_B}, E_{R1_C}, F = 011**

9. (7 points) For the architecture shown, write the sequence of signals and control actions necessary to implement the fetch cycle.



Cycle	Concrete RTL	Signals
1	**MAR ← PC**	**C_{MAR}, E_{PC_B}, M_ALU = 1**
2	**PC ← PC + 1, IR ← M[MAR]**	**C_{PC}, M_PC = 1, Read, M_MBR = 0, C_{IR}**

10. (15 points) A RISC processor executes the following code. There are data dependencies but no internal forwarding. A source operand cannot be used until it has been written. Assume that the first instruction begins executing in the first cycle.

a. (5 points) Assuming a four-stage pipeline (fetch (IF), operand fetch (OF), execute (E), operand write (W)), what registers are being read during the seventh clock cycle and what register is being written**?** **r3 is being written (W) and r3 and r6 are being read (OF), r3 value will not be correct but r6 will**

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
MUL r0, r1, r2	IF	OF	E	W										
ADD r3, r1, r0		IF	OF	OF	OF	E	W							
LDR r1, [r2]			IF	IF	IF	OF	E	W						
ADD r3, r3, r6						IF	OF	OF	E	W				
ADD r6, r0, r7							IF	IF	OF	E	W			
STR r3, [r6]									IF	OF	OF	OF	E	W

b. (10 points) Assuming a six-stage pipeline: fetch (F), register read (O), execute (E), memory read (MR), memory write (MW), register write (WB), how long will it take to execute the entire sequence? **21 cycles**

	1	2	3	4	5	6	7	8	9	10	11
MUL r0, r1, r2	F	O	E	MR	MW	WB					
ADD r3, r1, r0		F	O	O	O	O	O	E	MR	MW	WB
LDR r1, [r2]			F	F	F	F	F	O	E	MR	MW
ADD r3, r3, r6								F	O	O	O
ADD r6, r0, r7									F	F	F
STR r3, [r1]											

	12	13	14	15	16	17	18	19	20	21	22
MUL r0, r1, r2											
ADD r3, r1, r0											
LDR r1, [r2]	WB										
ADD r3, r3, r6	O	E	MR	MW	WB						
ADD r6, r0, r7	F	O	E	MR	MW	WB					
STR r3, [r1]		F	O	O	O	O	E	MR	MW	WB	