Please show **all** your work! Answers without supporting work will not be given credit. Write answers in spaces provided. Whenever you are required to write assembly code you may use **only** instructions from the supplied MIPS Instruction Reference. You have **2 hours** to complete this exam. The maximum number of points is **100**.

Name: ______________________ Student Number: ______________________

1 Multiple-Choice Questions (10 Points)

Correct answer: +1 Point, Wrong answer: -1 Point, No answer: 0 Points. Negative total points will be elevated to 0.

(a) When converting a two's-complement number from a 16 bit representation into 32 bits, the top 16 bits of the new 32 bit number will be all zero.
○ True ○ False

(b) Instruction pointer (PC) is used to indicate the location of the current frame.
○ True ○ False

(c) The control unit in a multicycle processor is implemented as a finite state machine.
○ True ○ False

(d) A multiplexor with a 32-bit output can be implemented as a collection of 32 multiplexors with 1-bit outputs.
○ True ○ False

(e) In two-way superscalar processor with 6 pipeline stages 12 instructions can be active at the same time.
○ True ○ False

(f) In a processor with 'In-order issue with in-order completion' policy an instruction can never be completed before an instruction which is located earlier in the program order.
○ True ○ False

(g) SRAM is used for caches due to its higher density compared to DRAM.
○ True ○ False

(h) In a direct mapped cache, a memory block can be mapped to any cache block.
○ True ○ False

(i) A 16 bit wide bus with a frequency of 200 MHz has a transfer rate of 100 MB/s.
○ True ○ False

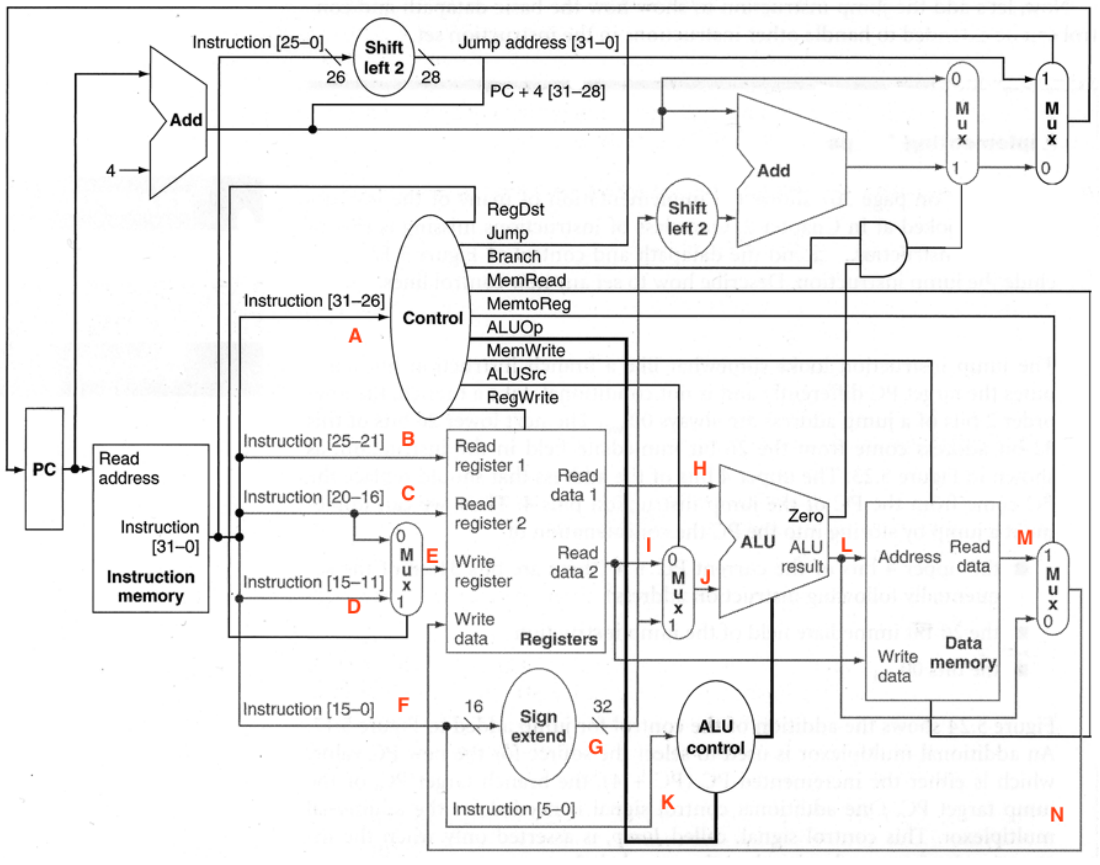
(j) Superpipelined processors have longer instruction latency than the Superscalar processors.
○ True ○ False

2 MIPS Single-Cycle Datapath (16 Points)

Consider the single-cycle implementation of a MIPS processor given below. Suppose that the processor executes the instruction `add $r20,$r22,$r23`. The `$PC` register holds `0x1004’0004`. The state of the register file is given in the following tables:

Program Counter
0x1004’0004

Register file	
Register	Content
$r20	0x0000’0035
$r21	0x0000’1042
$r22	0x0000’1040
$r23	0x0000’1018



Based on the given implementation and register/data states, answer the following questions. **Note:** For all the following tasks the `0x` prefix is used to indicate hexadecimal values.

(a) (5 points) How is the instruction `add $r20,$r22,$r23` encoded in the instruction memory? State the binary representation of this instruction and document your solution process. The opcode of `add` is `0x00`, function code is `0x20`.
`Hint:` `add` is an R-type instruction, the syntax of a `add` instruction is `add $rd, $rs, $rt`, the operation performed is `$rd = $rs + $rt`

(b) (4 points) What are the values of the control signals `RegDst`, `Jump`, `Branch`, `MemRead`, `MemtoReg`, `MemWrite`, `ALUsrc` and `RegWrite`? Also state "Do not Care" cases, if any (with "X").

RegDst	**Jump**	**Branch**	**MemRead**
MemtoReg	**MemtoReg**	**ALUsrc**	**RegWrite**

(c) (7 points) For each letter in the diagram, state the value of the signal on the corresponding wire. You may use binary, hexadecimal or decimal values. All (if any) undefined signals are to be marked with "X".

A	**B**	**C**	**D**
E	**F**	**G**	**H**
I	**J**	**K**	**L**
M	**N**		

3 Memory & Performance (12 Points)

(a) (4 points) Given a processor with a clock rate of 800 MHz which executes the following instruction-mix:

ALU:	60%	2.5ns
LOAD:	10%	6.25ns
STORE:	10%	5ns
BRANCH:	20%	3.75ns

Calculate the ideal CPI of the processor (without taking stalls through memory access into account).

(b) (4 points) Calculate the effective CPI of processor from (a) with only a L1 cache. Assume that a main memory access takes 80 clock cycles. The miss rate of the instruction cache is 2%, the miss rate of the data cache 5%. If you could not solve **(a)** assume the ideal CPI is 2.

(c) (4 points) We add a Level 2 cache with an access time of 25 ns. The miss rate of the L2 cache is 0.2%. What is the effective CPI of this system? If you could not solve **(a)**, assume the ideal CPI is 2.

4 MIPS (12 Points)

(a) (6 points) Consider a pseudo-instruction named `copge` with the syntax:
`copge $d, $s, $t`
which sets the register `$d` the maximum value from registers `$s` and `$t`. How could the assembler translate the code below using the basic Instruction Set?

```
copge $s3, $s5, $s2
```

(b) (6 points) Translate the following C code into an assembly code using instructions from the supplied MIPS Instruction References:

```
a[3] = a[1] - a[2];
if (a[3] != b)
        a[3] = 0
```

where `a` is an array of integers with its base address stored in `$s1` and the value of `b` is stored in `$s2`.

5 Pipelining & Hazards (20 Points)

(a) (6 points) Consider the assembly code below. We use a simple pipelined processor with simultaneous **R/W** operation (at the same clock cycle). The timing table of the instructions are drawn below. Indicate which instructions cause data hazards.

	instruction	1	2	3	4	5	6	7	8	9	10	11	12
1	lw \$s2 4(\$s5)	IM	ID	EX	DM	WB							
2	lw \$s3 8(\$s5)		IM	ID	EX	DM	WB						
3	add \$s1 \$s2 \$s3			IM	ID	EX	DM	WB					
4	add \$s2 \$s3 \$zero				IM	ID	EX	DM	WB				
5	add \$s3 \$s1 \$zero					IM	ID	EX	DM	WB			
6	sw \$s2 4(\$s5)						IM	ID	EX	DM	WB		
7	sw \$s3 8(\$s5)							IM	ID	EX	DM	WB	

example: line 1 and 3 because of register `$s2`

(b) (4 points) Which hazards can be resolved by forwarding? Which hazards can only be eliminated by stalling? Briefly explain your answer.

(c) (10 points) Consider the assembly code below.

```
1  addi  $s1, $zero, 1
2  LOOP: beq $s1, $zero, END
3        addi $s1, $s1, -1
4        j LOOP
5  END:
```

We use a pipelined processor **with** forwarding, with simultaneous **R/W** operation (at the same clock cycle). It supports **ID** stage jumps and the branch decision hardware is located at the **DM** stage (no branch prediction). How many cycles does the code require to run? Consider the code finished when the instruction on line 5 reaches the **IM** stage.
(tip: draw a timing table for at least the first loop)

instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

6 C Programming (10 Points)

(a) (4 points) What is the output of the following C program?

```
#include <stdio.h>

int main() {
    int i;
    for(i=0;i<5;i++) {
         int i=10;
         printf(" %d",i);
         i++;
    }
    return 0;
}
```

(b) (6 points) What is the output of the following C program? If the output is not "John Muster 22", how would you modify the `assignValues` function and the corresponding function call to make it so?

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct {
        char *firstName;
        char *lastName;
        unsigned int age;
} Person;

void assignValues(Person p, const char *firstName, const char *lastName, unsigned int age) {
        p.firstName = firstName;
        p.lastName = lastName;
        p.age = age;
}

int main() {
        Person p;
        p.firstName = Jane;
        p.lastName = Doe;
        p.age = -1;
        assignValues(p, "John", "Muster", 22);
        printf("%s %s, %i\n", p.firstName, p.lastName, p.age);
        return EXIT_SUCCESS;
}
```

7 Cache (10 Points)

(a) (3 points) Consider a directly mapped cache with size of 8 words and block size 1. The data addresses are accessed in the following sequence: `1 9 17 1 9 17 1 9 17`.
Describe shortly what happens in the cache (you do not need to draw the cache layout). How many misses will occur? Are these compulsory, capacity or conflict misses?

(b) (4 points) We have a 2-way set associative cache that is 8KB, with 4-word cache lines. What is cache line size? What is the size of the offset, index and the tag?

(c) (3 points) The table below contains the sequence of data addresses accessed by a program. Assume we use the (initially empty) cache from **(b)**. Fill in the table with **Hit** or **Miss**.

Address	Accesss Type
`0x1004`	
`0x1005`	
`0x1006`	
`0x2004`	
`0x3000`	
`0x3001`	

8 Branch Prediction (10 Points)

(a) (3 points) What is the purpose of the branch branch target buffer (BTB)?

(b) (4 points) Consider the MIPS code below.

```
1 addi  $t0, $zero, 0
2 addi  $t1, $zero, 15
3 LOOP: beq $t1, $zero, OUT
4        addi $t0, $t0, 1
5        sub $t1, $t1, $t0
6        j LOOP
7 OUT:
```

A one-bit branch predictor is used which is initialized at **taken**. What will be the predictions for the instruction `beq` at line 3? How many of the predictions will be correct?

(c) (3 points) For the code in **(b)** would a two-bit predictor initialized to taken make fewer mispredictions? Explain your answer.