THE AUSTRALIAN NATIONAL UNIVERSITY

Final Examination – June 2022

COMP/ENGN2219
Computer Architecture and Simulation
Final Examination

Reading Period: 15 minutes

Time Allowed: 180 minutes

Permitted Materials: Open Book

Total Marks: 100

Answer all questions.

Please ensure that you have carefully read the instructions provided in the Git repository's README file.

Good luck!

Question 1 [20 marks]

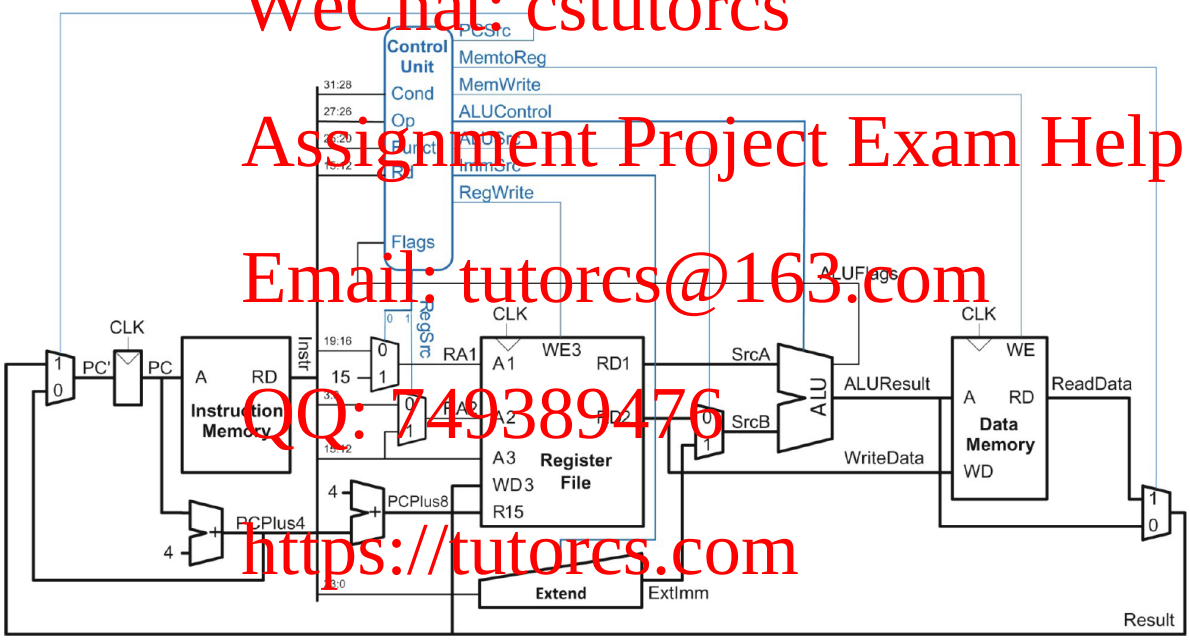
Provided below is a C code snippet, an ARM CPU microarchitecture diagram, and the delays of various logic elements.

```
int array[64];
int i, j, k;

for (i = 1; i < 64
    array[i] = 2;

for (j = 1; j < 32
    array[j] = arr

for (k = 32; k < 64; k = k + 1)
    array[k] = array[k] + 32;
```

Timing Values: $t_{cq\text{-}pc} = 25$ ps, $t_{mem} = 150$ ps, $t_{dec} = 70$ ps, $t_{mux} = 25$ ps, $t_{RFread} = 100$ ps, $t_{ext} = 30$ps, $t_{ALU} = 120$ ps, $t_{RFsetup} = 46$ ps

(a) [10 marks]

Translate the given C code to ARM assembly. Note that there are multiple ways to translate a given piece of C code to assembly. We are only concerned with correctness of your solution and not with code size or performance. A correct translation from C to assembly will receive full marks. The variables i, j, and k are mapped to ARM registers `R1`, `R2`, and `R3`. Assume the array base address is already in `R0`.

(b) [5 marks]

Given the timings and architecture diagram, determine the time to execute the following instructions: `LDR`, `ADD`, `BL`

(c) [5 marks]

Suppose your assembly code is executed on the single-cycle CPU above. Find the time it takes to execute the assembly code.

Question 2 [20 marks]

Design an FSM with one input, X, and two outputs, A and B. A should be 1 if X has been 1 for at least three cycles altogether (not necessarily consecutively). B should be 1 if X has been 1 for at least two conse our state transition diagram, encoded state transition table, next state and d schematic.

Submit your answer a name Q2 - eg Q2.jpg. Please ensure the submission is easy to read.

Question 3 **[20 marks]**

In this question we look at memory locality properties of matrix computation. The following code is written in C, where elements within the same row are stored contiguously. Assume each word is a 32-bit integ

```
for (I=0; I<8; I++
  for (J=0; J<8000
    A[I][J]=B[I][0
```


(a) [2 marks]

How many 32-bit integers can be stored in a 16-byte cache block?

(b) [4 marks]

In the above C code, references to which variables exhibit temporal locality?

(c) [4 marks]

In the above C code, references to which variables exhibit spatial locality?

(d) [2 marks]

How many 16-byte cache blocks are needed to store all 32-bit matrix elements being referenced?

(e) [4 marks]

Locality is affected by both the reference order and data layout. The same computation can also be written below in MATLAB, which differs from C by storing matrix elements within the same column contiguously in memory.


```
for I=1:8
  for J=1:8000
    A(I,J)=B(I,0)+A(J,I);
  end
end
```


In the above MATLAB code, references to which variables exhibit temporal locality?

(f) [4 marks]

In the above MATLAB code, references to which variables exhibit spatial locality?

Question 4 [20 marks]

Consider the ARM assembly code below. `func1`, `func2`, and `func3` are non-leaf functions. func4 is a leaf function. The code is not shown for each function, but the comments indicate which registers are used wit u may assume that the functions do not use any stack space unless required store any local data on the stack.

```
0x00091000 func1 ...                    -R10
0x00091020 BL func
. . .
0x00091100 func2 ...                    -R5
0x0009117C BL func
. . .
0x00091400 func3 ... ; func3 uses R3, R7-R9
0x00091704 BL func4
. . .
0x00093008 func4 ... ; func4 uses R11-R12
0x00093118 MOV PC, LR
```


(a) [5 marks]

Assume all instructions are 4 bytes. How many words are the stack frames of each function?

(b) [10 marks]

Sketch the stack after func4 is called but before the first instruction in func4 has executed. Clearly indicate which registers are stored where on the stack and clearly mark each of the four stack frames. Give register values where possible. Identify and label the address of each stack word in memory (use one column for memory addresses and another column for data similar to lecture slides). Show the stack word that SP points to in your stack illustration. Assume the stack frame of func1 starts at `0xBFFFFF00`. Submit your answer to this part as an image with the name Q4B - eg Q4B.jpg. Please ensure the submission is easy to read.

(c) [5 marks]

Most microprocessors today have separate caches for storing program instructions and data. This question is about the behavior of an instruction cache.

In this question, Assume:

- 64 KiB direct-mapped instruction cache with a 32-byte block.
- A full cache block can be fetched in one read operation.
- All instructions are 4 bytes.
- The above functions do not contain any branches or loops, except for those indicated.

What is the miss rate for the address stream of the program above? How is this miss rate sensitive to the size of the cache or the working set? How would you categorize the misses this workload is experiencing, based on the 3C model?

Question 5 [20 marks]

(a) [5 marks]

Given four prin

- Simplicity
- Smaller is
- Good desig promises.
- Make the

Evaluate the QuAC ISA against these principles, explaining your reasoning.

(b) [5 marks]

A QuAC CPU has made its way to space onboard a satellite, and you're in charge of keeping it running. Cosmic radiation has damaged the CPU, and it can no longer perform the `sub` instruction. Perform the operation `r3 := r1 - r2`; Given input values in `r1` and `r2`, subtract them and place the result in `r3`. You can use any of the registers as required, and any instructions other than `sub`.

(c) [5 marks]

QuAC's conditional execution only uses the Z, or Zero, flag. Explain how you can write a QuAC program to conditionally execute based on the Carry flag, using only the existing ISA (with no changes), and provide a brief assembly example.

(d) [5 marks]

Translate the following C code into valid QuAC assembly.

```c
short array[8] = {1,2,4,10,22,58,233,827};

void main(){
    short* arr_ptr = &array;
    for (short i = 0; i < 8; i++) {
        *(arr_ptr + i) = *(arr_ptr + i) - i;
    }
}
```