COL 216 Minor I

Sarang Chaudhari

TOTAL POINTS

34.5 / 40

QUESTION 1

1 Q1 10 / 10

✓ + **2 pts** **Register address is constant in an instruction. Array indices are usually not constants in a program.**

✓ + **4 pts** **Can't use registers for accessing array elements 'a[i]', if array index 'i' is calculated at run-time**

+ **4 pts** If array indices are constants, then we can use registers for storing array elements as 5 individual scalar variables

+ **0 pts** Incorrect/Incomplete

+ **4 Point adjustment**

Almost correct. If array indices in the program are constants, it is possible to store array as 5 independent register variables.

QUESTION 2

2 Q2 6 / 10

✓ + **2 pts** **Part a: case, address of first instruction of case mapping**

+ **0 pts** Part a: Incorrect answer

✓ + **4 pts** **Part b: Less space wastage**

+ **0 pts** Part b: Incorrect Answer

+ **4 pts** Part c: Hash computation time added

✓ + **0 pts** **Part c: Incorrect answer**

+ **2 pts** Part c: Correct answer along with Incorrect answer

+ **2 pts** Part b: Correct answer along with Incorrect answer

+ **1 pts** Part a: Incomplete answer

Part c: Computing the hash function may take too much time.

QUESTION 3

3 Q3 10 / 10

+ **0 pts** Incorrect

✓ + **5 pts** **Efficiency of the code**

✓ + **5 pts** **ori/addu instruction used**

+ **3 pts** Code not efficient

QUESTION 4

4 Q4 4.5 / 6

✓ + **3 pts** **IPC**

+ **1.5 pts** Incomplete Part a.

+ **0 pts** Wrong IPC

+ **3 pts** Clock Period

✓ + **1.5 pts** **Incomplete Part b.**

+ **0 pts** Wrong CP

+ **2 pts** Incomplete overall answer

+ **0 pts** Click here to replace this description.

a) IPC Omits complex instructions.
b) The clock period alone doesn't consider the amount of work done in one clock cycle.

QUESTION 5

5 Q5 4 / 4

✓ + **4 pts** **Opcode + function bits. Only opcode is also Ok.**

+ **0 pts** Incorrect

Department of Computer Science and Engineering, IIT Delhi
Computer Architecture (COL 216)
II Semester 2019-20
Minor Exam I
05 February 2020, 8:00 to 9:00

Name: Sarang Chaudhari	**Entry Number:** 2018CS10381

Write only in the space provided below the question.

1. **[10 Marks]** Can the MIPS register file be used to store an array of 5 integers? Justify.

An array has 2 properties:

~~An ordered~~
a) ~~set of elemen~~

a) An ordered list of elements : We can store 5 different integers of the array in 5 different registers (say for ex. \$s0 – \$s4) and address each one of them using the names of the registers (for this we will need to remember the 5 registers). ~~a~~ ~~Aso~~ Also any called procedure will not change these values as 5 registers are preserved by the callee. So we could store the integers in that sense. But ...

b) Continuous memory locations (hence they could be adderessed using the address of initial element or the pointer) : We cannot iteratively ~~iterate~~ work upon the numbers stored in the registers (for eg. in a loop) ~~nor~~ neither can be address ~~the~~ all the elements in some sense by using any of pointer or memory address as such. Hence this property cannot be implemented and therefore we 'cannot' store an 'array' of 5 integers in MIPS register file ('cannot' is quoted because even though we technically can, some properties won't hold).

Name: Sarang Chaudhari	Entry Number: 2018CS10381

2. **[2+4+4=10 Marks]** There was a suggestion in the class to use the HASH TABLE data structure to implement the C switch construct. Instead, we used a simple array to implement it.
   a. Explain how we could use the hash table to implement the switch.
   b. What is a possible advantage of using the hash table over an array to implement the switch?
   c. What is a possible disadvantage of using the hash table to implement the switch?

a. We would need to define some kind of a hashing function like a simple one ~~and~~ which manipulates the bits of the case statements. These ~~hases~~ hashes can inturn ~~brused~~ be used to maintain a Hash Table containing addresses of case instructions (similar to the array implemantation).

b. An advantage is that we won't need ~~an~~ a ~~arry~~ large array for storing just 2 cases (like 0 & 100000). Efficiently implemented hash table will reduce the total Space required.

c. We know that hashing doesn't necessarily have to be an one-one bijection. One could map the integers to a smaller set of hash values to reduce the space. This will definitely result in a collision of 2 entries. Now first of all, we won't be able to store 2 addresses in a single entry of the table if we ~~ars~~ are storing the Table like an array. Or if we could using any other approach, we will need to check both the cases in an entry and hence defeat the purpose of constant time look-up in a Switch statement.

Name: Sarang Chaudhari	Entry Number: 2018CS10381

3. **[10 Marks]** We would like to store a 32-bit value in MIPS register $s3. Write an instruction (or sequence of instructions) to do this. The sequence should be as short as possible. You can use the `lui` instruction that loads a 16-bit immediate value into the upper 16 bits of a register, setting the lower 16 bits to zero (its format is: *lui <register>, <constant>*)

We know that for an I-type instruction, we ~~can~~ only have 16 bits for the immediate constant. Hence we need to store it in 2 steps.

```
1.      lui   $s3 ,  (upper_16_bits)
2.      ori   $s3 ,  ( lower_16 bits)
```

As given, we have used lui to initialise the upper 16 bits of the constant. Now we have lower 16 bits as 0 in the register $s3. For appending the lower_16_bits we use ori as it does an unsigned extension of the 16 bits to 32 bits (as opposed to addi which does sign extension). Hence finally we will have

```
      upper_16_bits   0000 0000 0000 0000
or    0000 0000 0000 0000   lower_16_bits

   =  upper_16_bits lower_16_bits  =  complete_32_bits.
```

Name: Sarang Chaudhari	Entry Number: 2018CS10381

4. **[3+3=6 Marks]** Explain why each of the following, by itself, is NOT a good enough measure of computer system performance:
   a. Instructions Per Cycle (IPC)
   b. Clock Period

a. Let us consider that we convert the given program into a program with much basic instructions than MIPS. But due to this, its program sizeis 5x the program size in MIPS. Because the instructions are simpler, we could execute more (2x) instructions per cycle (~~But~~ Hence 2x IPC). But the overall performance is hampered by 2.5 times and hence IPC is not a good enough measure of system performance.

b. Decreasing the clock period ~~decreases~~ & increases the total number of clock cycles in a given time, no doubt. But due to shorter clock period, we may need to schedule some instructions to the next clock cycle, as all of them may not fit is a single clock. Decreasing the clock period can increase the performance in some cases uptil a point, but it may even hamper its performance. Hence clock period alone is also not a goodmeasure of performance.

Name: Sarang Chaudhari	Entry Number: 2018CS10381

5. **[4 Marks]** How does the MIPS processor know whether an instruction to be executed is of R-type, J-type, I-type, etc.?

The first 6 bits of any instruction gives us the opcode. This ~~uniquely~~ ~~ad~~ identifies the diferent types of instructions (not uniquely eg. add, sub, sll has the same initial 6 bits (opcode), but the opcode at least distinguishes between the 3 classes of intructions). This is possible because any kind of instruction reserves its initial 6 bits for its identification and the rest division depends upon the type of instruction it is.