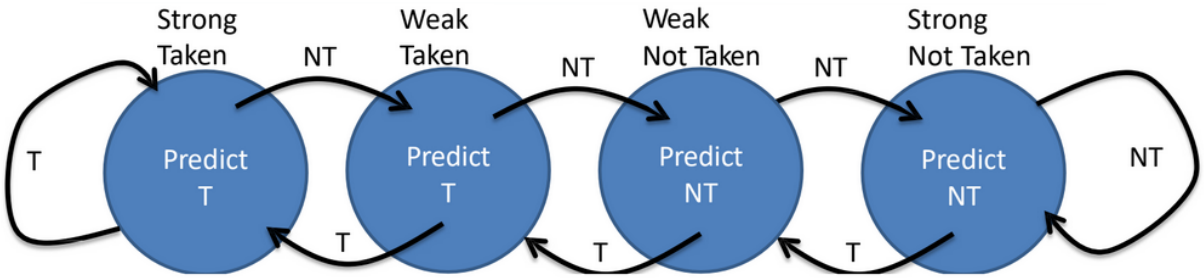
CSE/ECE 511: Computer Architecture

Time: 2 Hours **End Semester Examination** **[Marks: 17 + 18 + 10 + 20]**

Instructions:

1. ***Institute's plagiarism policy will be followed in case of any unfair means observed.***
2. ***Use of Gadgets such as Mobile phones, watches (analog and digital), tablets etc. are strictly prohibited in the exam hall.***
3. ***Use of ONLY non-programmable Scientific Calculator is allowed.***

Q1. There are N 2-bit saturating counter branch predictors in a CPU. All these predictors are independent of each other. The 4 least significant bits of a branch instruction address are used to identify the predictor for that instruction. Each predictor follows the same finite state machine given below:



Assume that all the predictors are initially in the *strong not taken* state.
The following code is then executed on the CPU.

Initially, all the registers have the value of 0 stored in them

```
str R1, 0x00000000;          // Store a 32-bit constant value of 0 into register R1
str R2, 0x00000000;          // Store a 32-bit constant value of 0 into register R2
loop:
        add R12, R13, R14;           // Add R13 and R14 and store the result in R12
        sub R15, R10, R11;           // Subtract R11 from R10 and store the result in R15
        inv R2;                      // Bitwise flip all bits of R2
        bneq R2, 0x00000000, endif;  // Branch to label endif if R2 != 0x00000000
                mul R11, R12, R13;   // Multiply R12 and R13 and store the result in R11
        endif:
        add R1, R1, 0x00000001;      // Add 1 to R1 and store the result back into R1
        bneq R1, 0x00000005, loop;   // Branch if R1 != 0x00000005 to the label loop
```

The address of the branch instruction in **bold** is `0x1584e4f9`

The address of the branch instruction in *italics* is `0x9827858b`

(Note: 0xabcd represents a hexadecimal value)

a. What is the value of N, i.e. how many total 2-bit branch predictors are there in the CPU? **[1 Mark]**

b. Compare this setup of N predictors with a single 2-bit saturating counter global branch predictor. What are the benefits and drawbacks of using N predictors instead of a single global predictor? **[1+1 Marks]**

c. Identify the index of the predictor corresponding to the two branches. **[1 Mark]**

d. For each branch in the program, identify the state of the N branch predictors and calculate the overall accuracy. Fill the following table in your answer sheets for this question. You may add more rows and columns if you wish to do so. **[11 Marks]**

Instruction Address	Prediction	Actual state of the branch (taken or not taken).	Was the prediction accurate?	New state of the N predictors

e. Compute the overall prediction accuracy for the branch prediction system. **[2 Marks]**

Q2. Assume that you are given a CPU with an in-order ISA. This CPU will run the same compiled program executable multiple times with changes to the microarchitecture. Assume that the changes to the microarchitecture **do NOT have any effect** on the operating frequency of the CPU, i.e. the number of cycles per second. The CPU is *initially* connected directly to 4GB of main memory, i.e. RAM.

The program:
Note that the code mentioned below is single-threaded, and it is pinned to a specific core of the CPU. Also, note that the ISA initially supports a 64-bit load instruction which can load 64 bits at once. The compiled program uses this 64-bit load instruction.

```
// a, b, and c are 64-bit unsigned integer arrays of length N
uint64_t a[N], b[N], c[N];
// N is a large positive integer
for(int i = 0; i < N; i++)
{
      a[i]  = b[i] + c[i];
}
```

Answer whether the performance of the CPU (how fast the program executes) will increase or decrease with the modifications mentioned below. Note that each modification is independent of other modifications, i.e**.** the modifications **do NOT accumulate.** In other words, all parts are independent of each other.
→Write the reason for the increase/decrease of performance for the following modifications.

a. The RAM (memory) of the system is increased to 8GB (memory access latency remains the same).
b. Another CPU core is added to the processor.
c. A branch predictor is added to the processor.
d. A Cache is added between the processor and the RAM (memory).
e. The memory bus width is decreased to 32 bits, A different load instruction is added (which replaces the original 64 bit load) which can only load 32 bits of data at once. The program is compiled again to incorporate this new load instruction.
f. Full register bypassing support is added to the CPU.

→ *Be crisp and precise with the reasons (few words only).*
→ Fill the following table in your answer sheets for each of the parts (Do not mention the exact numerical value, just indicate an increase, decrease or no change, along with the reasoning): **[6×3 = 18 Marks]**

Microarchitecture changes	Instructions per program and reason	Cycles per instruction and reason	Effect on the execution time and reason
Adding more memory			
Adding another core			
Adding branch predictor			
Adding Cache			
32 bit support			
Full Register Bypassing			

Q3. Consider a 16-bit machine with a set-associative cache. The total cache size is 512 B. The associativity of the cache is 4. The length of the memory location is 1 byte, and the cache block (also known as cache line) size is 8 bytes.

a. Calculate the total number of sets present in the cache. **[1 Mark]**
b. Draw the 16-bit memory address showing the tag, index, and offset bits. **[2 Marks]**
c. Which memory locations will be brought from the main memory to the cache block if the 0x2C68 memory location is accessed? **[3 Marks]**
d. Consider a strided prefetch scheme that prefetches the data 32 locations after the 0x2C68 memory address. Determine the memory locations (main memory) from which the data will be accessed. **[4 Marks]**

Q4. Consider the following assembly code; the processor uses an in-order micro-architecture:

```
loop:  add r1 r2 r0
       sub r2 r3 r9
       add r3 r3 r0
       sw  r1 #10(r7)
       beq r1 r3 loop
```

Assume the initial values of registers are:

r0 = 1 ; r1 = 5 ; r2 = 9 ; r3 = 9 ; r9 = 1

a. Write down *all the possible* data hazards in the above code, their associated registers and the instructions. **[5 Marks]**
b. Draw the corresponding pipeline diagram assuming no bypassing. **[5 Marks]**
c. Report the final values of registers r0, r1, r2, r3 and r9 after the code has been executed. **[5 Marks]**
d. Rename the registers so that only true dependencies remain. Write the code with the correct **physical** registers. Assume that there are 10 architectural registers r0 to r9 which are mapped to 10 physical registers p0 to p9. Apart from these 10 physical registers, there are 10 more physical registers, p10 to p19, resulting in a total of 20 physical registers p0 to p19. **[5 Marks]**