

1) Consider the following code segment:

```
IF:   BEQ R3, R8, ELSE
      ADD R4, R5, R6      /* ADD should not be executed if the branch is taken */
      SUB R8, R5, R6
      .
      .
      JUMP END_IF
ELSE: OR R3, R3, R2
```

- During which stage is the target address (address of the “ELSE” label) likely to be calculated for the BEQ instruction?
- During which stage of BEQ instruction is the comparison between registers (R3 and R8) likely to be performed?
- If the branch is taken, then how many instructions will be in the pipeline that need to be thrown away?
- If the branch is not taken and we continue to fetch instructions sequentially, then there is a branch penalty of _____ cycles.

2) Suppose that you are writing a compiler for a machine that has opcodes to statically predict whether or not branches will be taken. For each of the following HLL statements, predict whether or not the compiler should predict taken or not. (Briefly justify your answer)

- | | | | |
|--------------------|--------------------|-------------------------|--------------------------------------|
| a) if (x > 0) then | b) if (x = 0) then | c) for i := 1 to 500 do | d) if (ch >= 'a' and ch <= 'z') then |
| end if | end if | end for | end if |

3) Answer the following questions about Branch-prediction buffer:

a) If the branch instruction is in the Branch-prediction buffer, will the target address supplied correspond to the instruction that should be executed next? (Briefly justify your answer)

b) What if the instruction is a branch instruction and it is not in the Branch-prediction buffer?

c) Should the Branch-prediction buffer contain entries for unconditional as well as conditional branch instructions?

4) Consider the following bubble sort algorithm that sorts an array numbers[1..n]:

```

BubbleSort (int n, int numbers[])
  int bottom, test, temp;
  boolean exchanged = true;
  bottom = n - 2;
  while (exchanged) do
    exchanged = false;
    for test = 0 to bottom do
      if number[test] > number[test + 1] then
        temp = number[test];
        number[test] = number[test + 1];
        number[test + 1] = temp;
        exchanged = true;
      end if
    end for
    bottom = bottom - 1;
  end while
end BubbleSort
    
```

Part (a) answer	Part (b) answer

- a) Where in the code would unconditional branches be used and where would conditional branches be used?
 - b) If the compiler could predict by opcode for the conditional branches (i.e., select whether to use machine language statements like: “BRANCH_LE_PREDICT_NOT_TAKEN” or “BRANCH_LE_PREDICT_TAKEN”), then which conditional branches would be "PREDICT_NOT_TAKEN" and which would be "PREDICT_TAKEN"?
 - c) Assumptions:
 - n = 100 and the numbers are initially in descending order before the bubble sort algorithm is called
 - the five-stage pipeline of the lecture
 - the outcome of conditional branches is known at the end of the E stage
 - target addresses of all branches is known at the end of the E stage
 - ignore any data hazards
- Under the above assumptions, answer the following questions:
- i) If fixed predict-never-taken is used by the hardware, then what will be the total branch penalty (# cycles wasted) for the algorithm? (Here assume NO branch-prediction buffer)

ii) If a branch-prediction buffer with one history bit per entry is used, then what will be the total branch penalty (# cycles wasted) for the algorithm? (Assume predict-not taken is used if there is no match in the branch-prediction buffer) Explain your answer.