Question 1 (30 points): For each of the following statements you have three options: (i) leave it blank; (ii) Mark it with **T** to indicate that the statement is true; (iii) Mark it with `F` to indicate that the statement is false. **You lose 3 points for each incorrect answer.** You win 6 points for each correct answer. You don't win or lose any points if you leave the statement blank. Regardless of how many statements you mark wrong, your score in this question cannot be below zero.

a. (**T**) In an embedded processor whose design was derived from the MIPS architecture, the address field for a branch instruction is 12 bits. The execution procedure for a branch is the same as in MIPS:

```
bne  $s3, $s4, 8:      PC <-- PC + 4
                       if($1 != $2) then PC <-- PC + (address << 2)
```

The largest distance that a branch can jump backward in this architecture is 2047 instructions.

The largest negative 12-bit address is 1000 0000 0000 = - 2^{11} = - 2048. The PC is incremented by 4 before this value is shifted and added to the PC, thus the largest jump backward with a branch is 2047 instructions.

The solution above was my solution and thus the answer should be TRUE. However someone pointed out during the exam that you could have a larger "backward branch" if your branch instruction is at a very high address, say address 0xFFFF FFF0 and you do a forward branch and wrap around. Although this is not in the spirit of the question, I have to accept the logic of this argument. Therefore, for grading purposes, I must also accept FALSE as a correct answer. If the question was left blank, it means that the student did not know how to solve it and thus the grade should not be affected. A tighter statement of the question would also state the address where the branch instruction is.

b. (**T**) Assume two cache designs C_A and C_B have the same block size. C_A is a 32 KB 2-way set associative cache and C_B is an 16 KB direct-mapped cache. The length of the tag, measured in the number of bits, is the same in C_A and in C_B.

C_A has twice as many blocks as C_B, but each index points to a set that contains two blocks. Thus both caches use the same number of bits for indexing and the same number of bits for offset (they have the same block size). Therefore their tags are also the same length

c. (**F**) The following code correctly executes an atomic swap between the value stored in the address specified by `$s1` and the content of register `$s4`:

```
try:        add   $t0, $zero, $s4
            ll    $t1, 0($s1)
            sc    $t1, 0($s1)
            beq   $t0, $zero, try
            add   $s4, $zero, $t1
```

The `sc` instruction should store the value of `$t0`, instead of `$t1`.

d. (**F**) The *sticky bit* is used in floating-point arithmetic to improve the performance of floating-point addition.

The sticky bit improves the *precision* of floating-point addition.

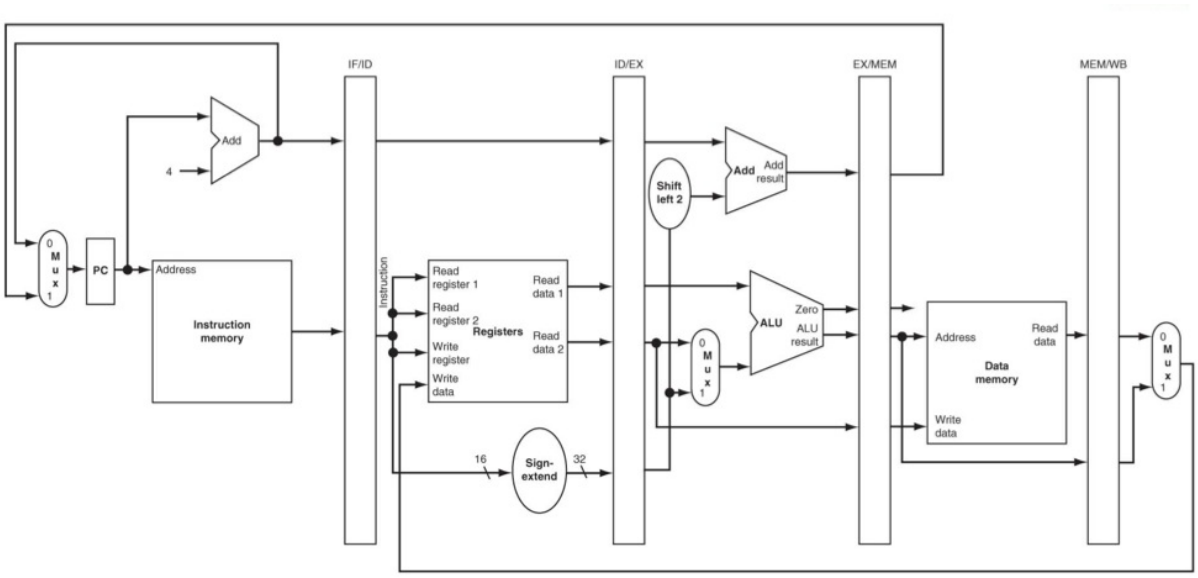

Figure 1: A design for a MIPS pipeline.

e. (**T**) Figure 1 depicts the architecture of a 5-stage pipelined implementation of the data path for the MIPS architecture. This drawing is incorrect because the instruction fetched at time T_i will write to the register specified by the instruction fetched at time T_{i+3}.