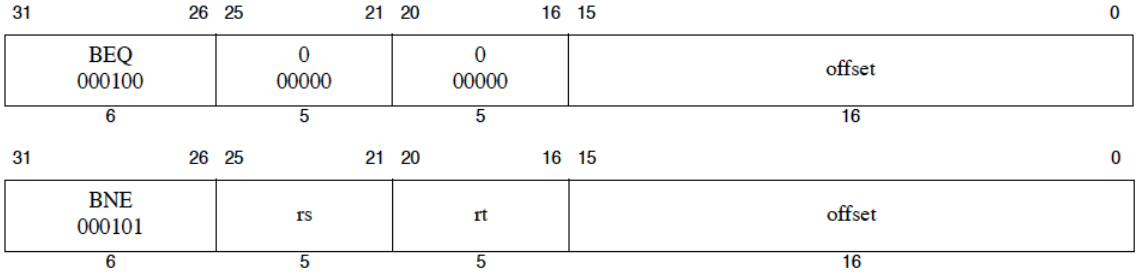
Question 1 (30 points):

You are working in a team that is writing a functional simulator for the MIPS architecture. One of the functions that needs to be written is called `BranchTarget`. It computes the target of a branch when the branch is taken.

Below are the specifications for the format of the `beq` and the `bne` instructions for the MIPS architecture. The operation of the `bne` instruction is described as follows:

PC $\leftarrow$ PC + 4
if rs $\neq$ rt then PC $\leftarrow$ PC + sign-extend(offset << 2)

Write the code for the `BranchTarget` subroutine that receives in `$a0` the address in which an instruction is fetched, and in `$a1` the 32-bit binary representation for the instruction. `BranchTarget` returns 0 if the fetched instruction is not a `bne` or a `beq` instruction. It returns the branch target, *i.e.* the value of the PC after the execution of the branch when the instruction is either a `bne` or a `beq` and the branch is taken. The code from `BranchTarget` must follow all the MIPS calling conventions.




```
BranchTarget:   srl  $t0, $a1, 26       # $t0 <-- instruction opcode
                li   $t1, 0x0004        # $t1 <-- beq opcode
                beq  $t0, $t1, IsBranch # if instruction is beq goto IsBranch
                li   $t2, 0x0005        # $t2 <-- bne opcode
                beq  $t0, $t2, IsBranch # if instruction is bne goto IsBranch
                add  $v0, $zero, $zero  # It is a non-branch instruction
                jr   $ra
IsBranch:       sll  $t3, $a1, 16       # $t3 <-- 0x offset 0000 0000 0000 0000
                sra  $t3, $t3, 14       # $t3 <-- sign-extend(offset << 2)
                addi $v0, $a0, 4        # $v0 <-- PC+4
                add  $v0, $v0, $t3      # $v0 <-- PC+4 + sign-extend(offset << 2)
                jr   $ra
```