1. According to the results of part 1, the vast majority of instructions with this architecture are load-instructions. This is presumably in part due to PISA, like MIPS, being a load/store architecture – all operations occur only between registers, and no operations can be performed on or from memory. This means there is poor code density and that it takes extra instructions to do the same thing.

When designing a new instruction set, I would implement a register-memory architecture. That way instructions can still be written as register-register so it's easy for the compiler to generate efficient code. However programs could be written to do operations directly through memory and avoid extra load instructions.

Additionally, floating point-instructions are used very little by the benchmarks other than fpppp. Unless I was developing an application-specific architecture heavily reliant on floating-point operations, I would strongly consider only supporting integer operations to save space and reduce the number of instructions.

2. The PC (program counter) stores the memory address of the next instruction to be executed. After fetching the instruction, the PC is incremented so that it stores the address of the next instruction. All instruction sizes in a RISC architecture such as MIPS are the same size, and thus the PC is incremented by the same quantity each instruction.

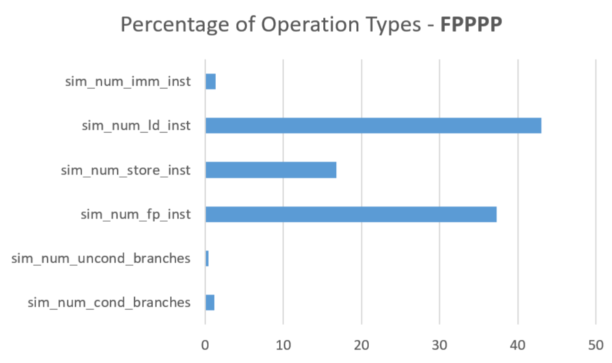
With MIPS64, all instructions are 32 bits/4 bytes and so the PC is incremented by four each instruction. Simplescalar has an extended 64-bit instruction encoding, and so increments PC by eight. This happens in the instruction cycle in the `sim_main` function at the following lines of code:

```
regs.regs_PC = regs.regs_NPC;
regs.regs_NPC += sizeof(md_inst_t); // 8
```

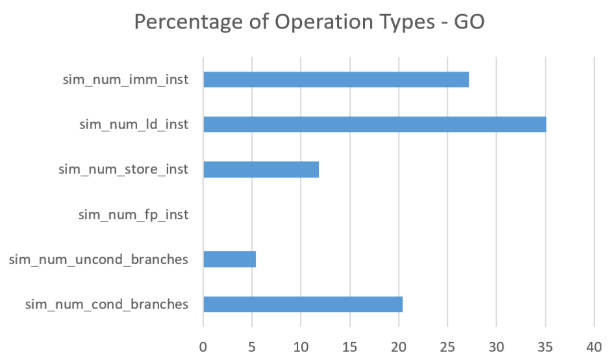
3. Every time a bit is flipped in a transistor it causes current to flow and consumes some dynamic power. This means the fewer bits are flipped, the less power is consumed. With shorter instruction encoding sizes, this means less bits must be flipped on average to change the instruction.

Since PISA has 64-bit instruction encoding, this means the same code executed on a PISA architecture would consume more power than that executed on a 32-bit instruction architecture like MIPS.

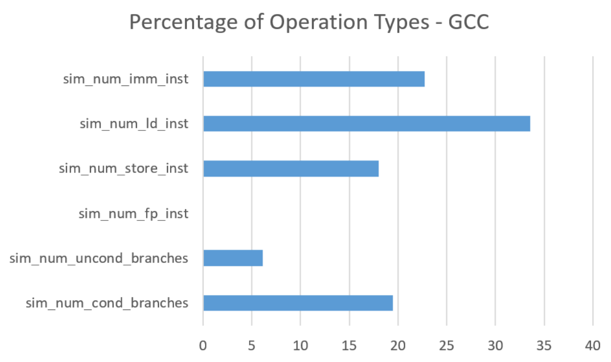
Additionally, when there are fewer total instructions made less bits have to be switched to execute a command. Since register-memory architectures don't have to execute as many load-instructions because they can operate on memory locations directly, they consume less power.



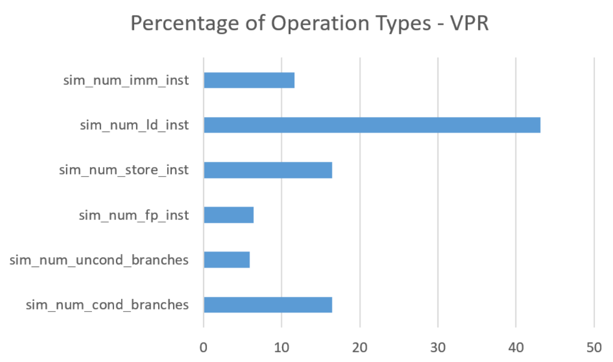
(a) fpppp



(b) go



(c) gcc



(d) vpr

Figure 1: Percentage of Executed Instruction Types by Benchmark

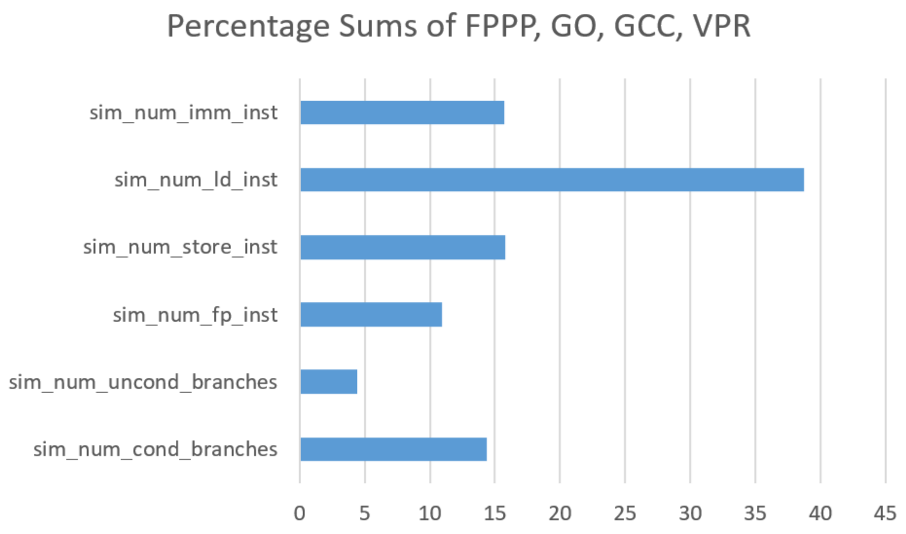
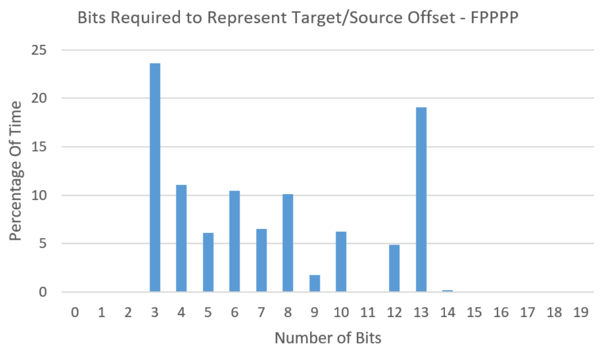
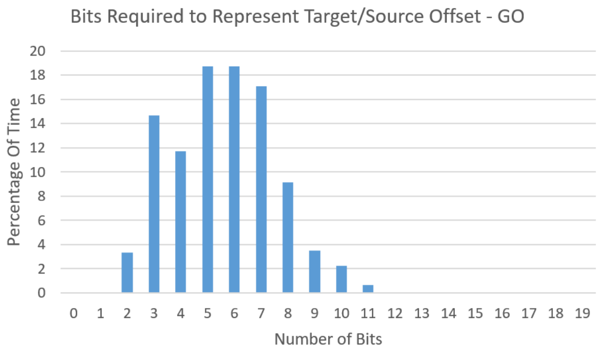


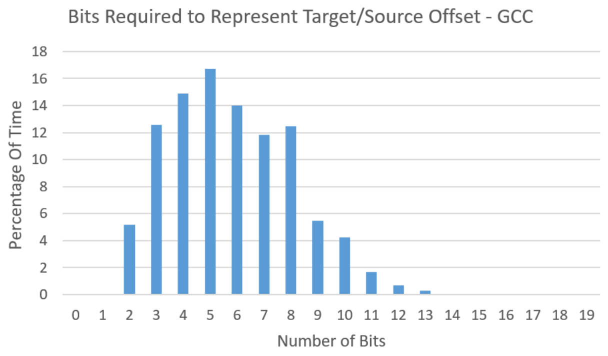
Figure 2: Percentage of Executed Instruction Types For All Benchmarks



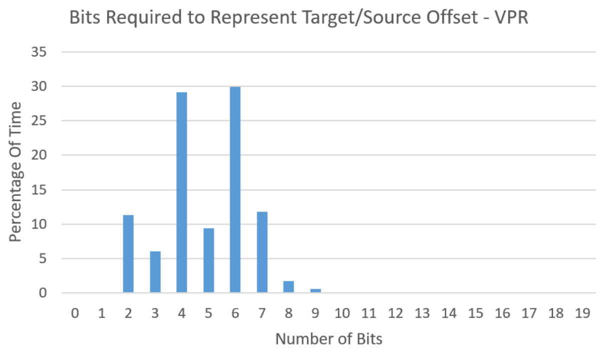
(a) fpppp



(b) go



(c) gcc



(d) vpr

Figure 3: Bits Required to Represent The Target/Source Offset of Conditional Branchs

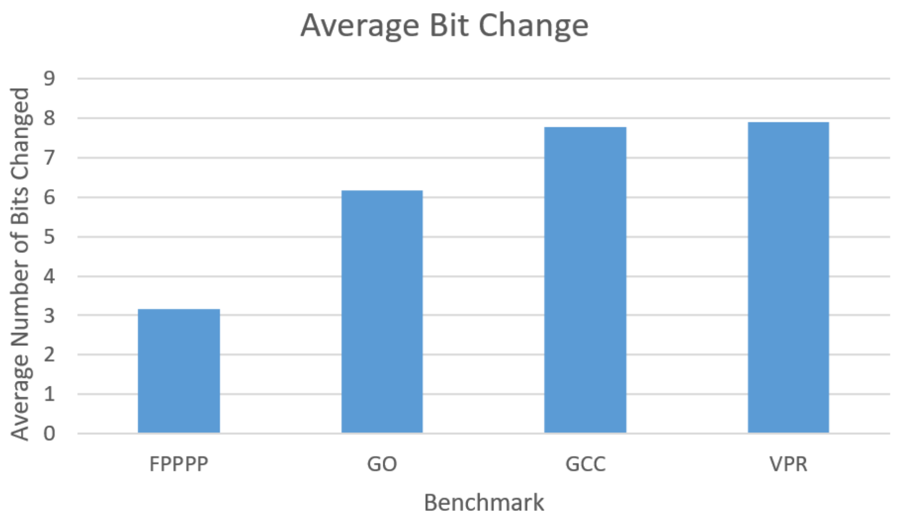


Figure 4: Average Bits Changed in a Register by Benchmark