**Discussions:**

**Problem Statement 1)**

\*Assuming that we are inspecting hardware differences rather than ISA

Looking at the 4 different benchmarks it can be seen each one has different requirements of the instruction set. The fpppp benchmark is the only benchmark that has a large amount of floating point instructions. Having specific hardware to increase the speed that these instructions are executed would speed up this program significantly but not the vpr, go, and gcc benchmark tests. As can be seen roughly 30% of the operations have to do with memory, therefore it is important to make sure the instructions for loads and stores are efficient and are intelligent such as making use of locality.

**Problem Statement 2)**

Instructions are 32 bits long which is why Simplescalar increments the PC by 8 bytes, which is 32 bits, to find the next instruction. In MIPS64 commands are word addressable which can be in common sizes of 8, 16, or 24 bits, therefore in MIPS64 sometimes the command increment will not be 8 bytes. Large offsets cause problems in locality due to a memory fetching multiple commands nearby each other to que up for the processor however when making a large jump this causes the advantages of locality to be unusable.

**Problem Statement 3)**

The more bits change, the more power a processor uses. Possibly making hardware that handles more operations to registers at the same time will avoid multiple accesses to the same register to change.

**Problem Statement 1 & 3)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **fpppp results:** |  |  | **gcc results:** |  |
| sim\_num\_cond\_branches | 0.0105 |  | sim\_num\_cond\_branches | 0.1522 |
| sim\_num\_uncond\_branches | 0.0036 |  | sim\_num\_uncond\_branches | 0.0478 |
| sim\_num\_floating\_point\_inst | 0.3321 |  | sim\_num\_floating\_point\_inst | 0 |
| sim\_num\_store\_inst | 0.1493 |  | sim\_num\_store\_inst | 0.1407 |
| sim\_num\_load\_inst | 0.3835 |  | sim\_num\_load\_inst | 0.2621 |
| sim\_num\_immediate | 0.0118 |  | sim\_num\_immediate | 0.1774 |
| sim\_avg\_reg\_bits\_chng | 2.998 |  | sim\_avg\_reg\_bits\_chng | 7.4573 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **vpr results:** |  |  | **go results** |  |
| sim\_num\_cond\_branches | 0.1088 |  | sim\_num\_cond\_branches | 0.1203 |
| sim\_num\_uncond\_branches | 0.0389 |  | sim\_num\_uncond\_branches | 0.0319 |
| sim\_num\_floating\_point\_inst | 0.0423 |  | sim\_num\_floating\_point\_inst | 0 |
| sim\_num\_store\_inst | 0.1092 |  | sim\_num\_store\_inst | 0.0696 |
| sim\_num\_load\_inst | 0.2856 |  | sim\_num\_load\_inst | 0.2067 |
| sim\_num\_immediate | 0.0771 |  | sim\_num\_immediate | 0.1601 |
| sim\_avg\_reg\_bits\_chng | 7.4062 |  | sim\_avg\_reg\_bits\_chng | 8.5629 |

**Problem Statement 2)**