Jay Chang
39787114
Oct 5 2015

CPEN 411 Computer Architecture
W1 2015

Assignment 1: Functional Simulation

Report Analysis

ANALYSIS OF PART ONE & TWO

For part one and two, we can see the data in Figure 1 and Figure 2 to Figure 5. From the first part when we analyzed the percentage of the executed instructions, I noticed that the load instruction is executed the most across all four benchmarks and that floating-point and unconditional branches are executed the least. In addition, for the second part, having an offset of 5 appears most frequently for all of the benchmarks. When designing new instructions set, from the data, we can assume that the load instruction will be used most frequently. From this, when implementing new instructions sets, we can find out which instructions will be used more and optimize those instructions by allocating resources or optimizing the hardware and code towards that section. Also by knowing the frequency of the offset, we can further optimize by creating our instructions set to have a lower offset. Since these offset tells us how much our code jumps, having a lower offset means that our code can run faster by having shorter jumps. So from using these data, we can determine where we can optimize our code.

ANALYSIS OF PART THREE

For part three, we can see the data in Figure 6 the average number of bits changed per instruction. Because some registers are used more than others, we can partition the registers in accordance to their usage. That way, all the most commonly used registers are always together and become more predictable when accessing the registers. We can save power when accessing the registers. Also, registers are just a bunch of D-flip flops grouped together. One way of optimizing this hardware side of it is to optimize the way the registers charge and discharge to save on the power so that the registers can themselves can be optimized for power saving and efficiency.

QUESTION FROM PART TWO

SimpleScalar PISA utilizes 64-bit encoding. The PC increments by 8 bytes, which in turn is 64 bits total. However, for MIPS, we are studying is a 32-bit instruction set version. So it is different. There are however 64-bits variants of MIPS that uses 64-bit wide instructions set.

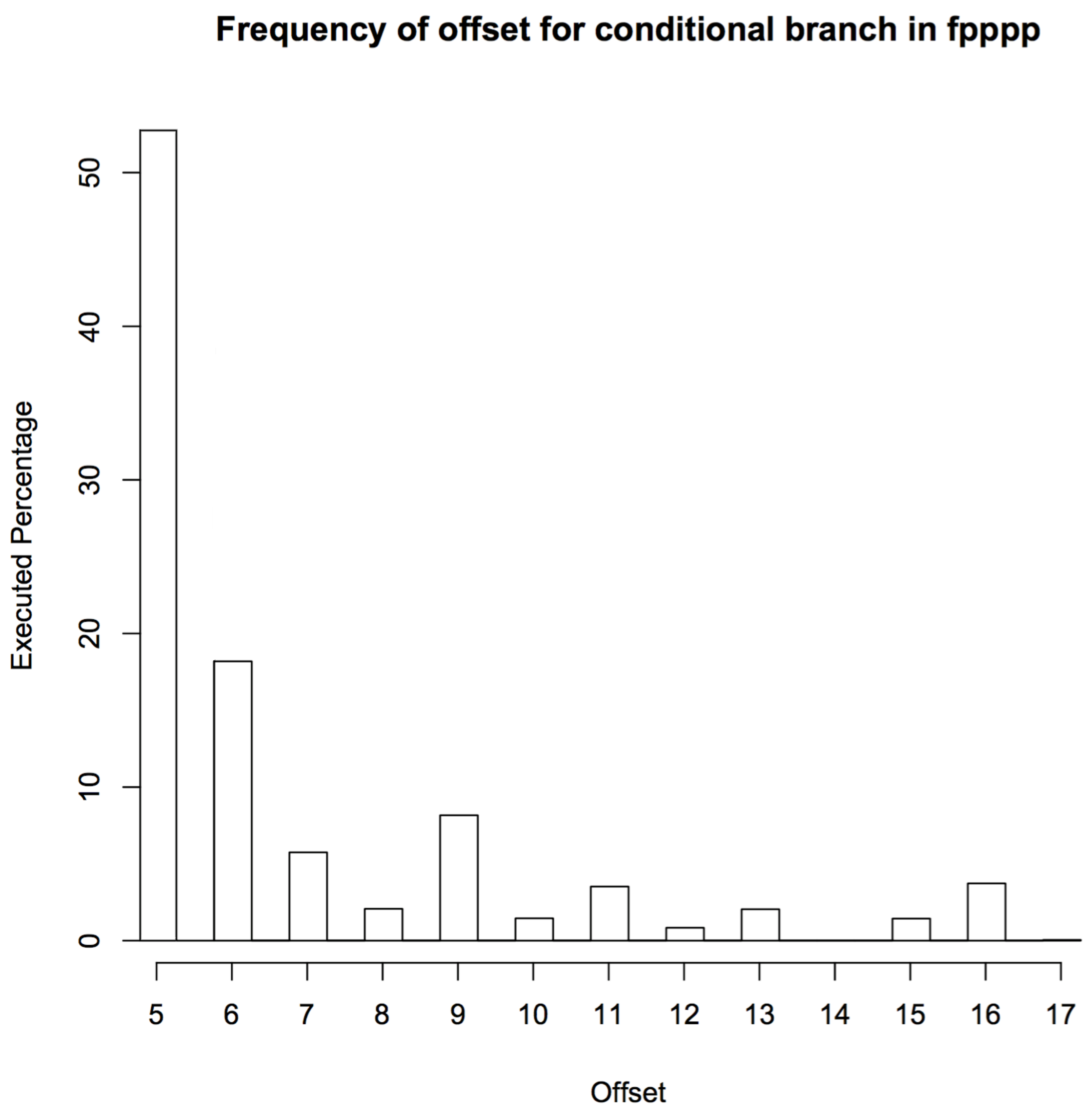
Jay Chang
39787114
Oct 5 2015

FIGURE 1.

fpppp	
Conditional Branches	1.05%
Unconditional Branches	0.36%
Floating-Point Instructions	0.31%
Store Instructions	14.93%
Load Instructions	38.35%
Instructions with Immediate Operands	1.17%
go	
Conditional Branches	12.03%
Unconditional Branches	3.19%
Floating-Point Instructions	0%
Store Instructions	6.96%
Load Instructions	20.67%
Instructions with Immediate Operands	16.01%
gcc	
Conditional Branches	15.22%
Unconditional Branches	4.78%
Floating-Point Instructions	0%
Store Instructions	14.07%
Load Instructions	26.21%
Instructions with Immediate Operands	17.74%
vpr	
Conditional Branches	10.88%
Unconditional Branches	3.89%
Floating-Point Instructions	3.43%
Store Instructions	10.92%
Load Instructions	28.56%
Instructions with Immediate Operands	7.71%

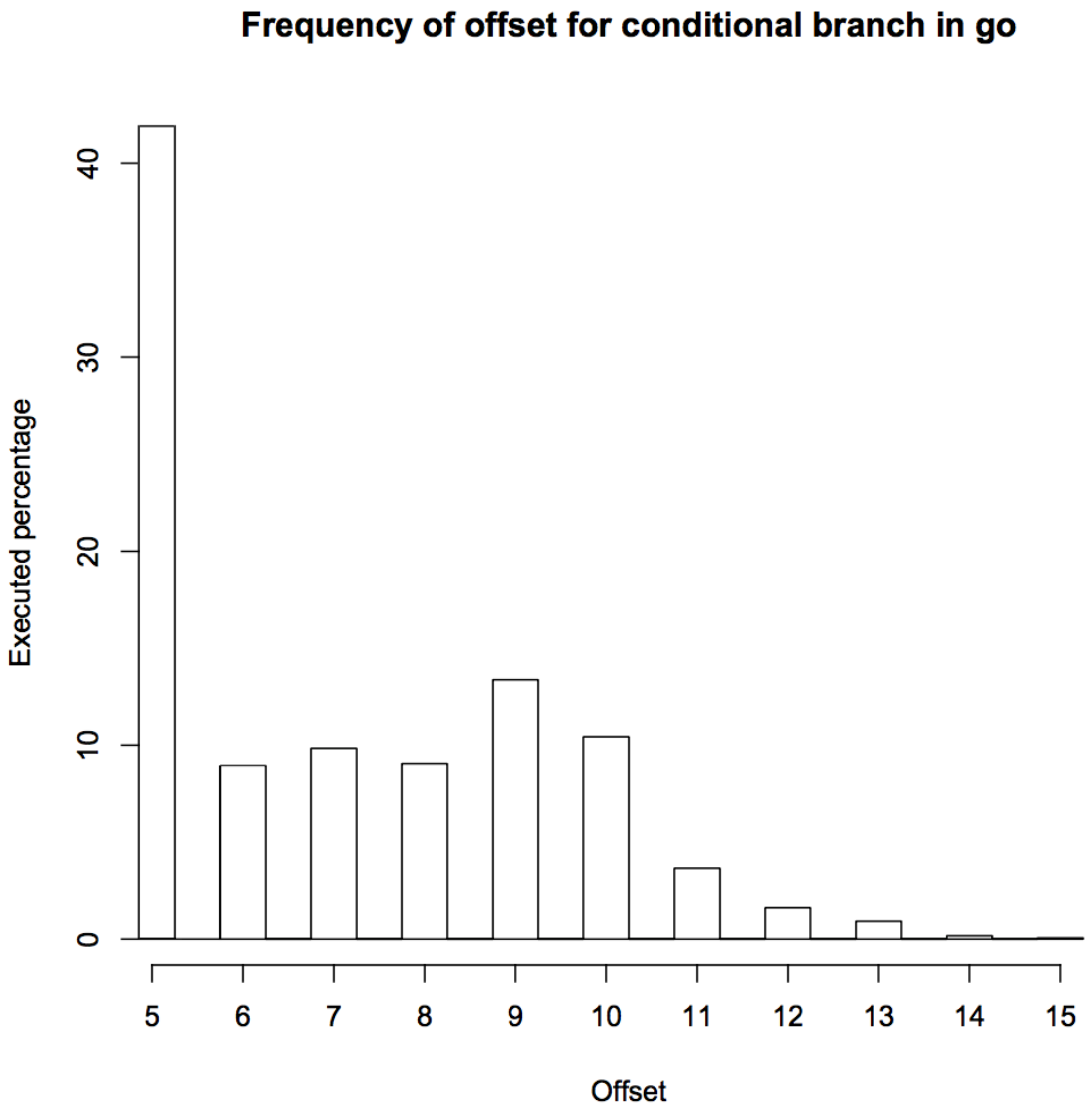
Jay Chang
39787114
Oct 5 2015

FIGURE 2.



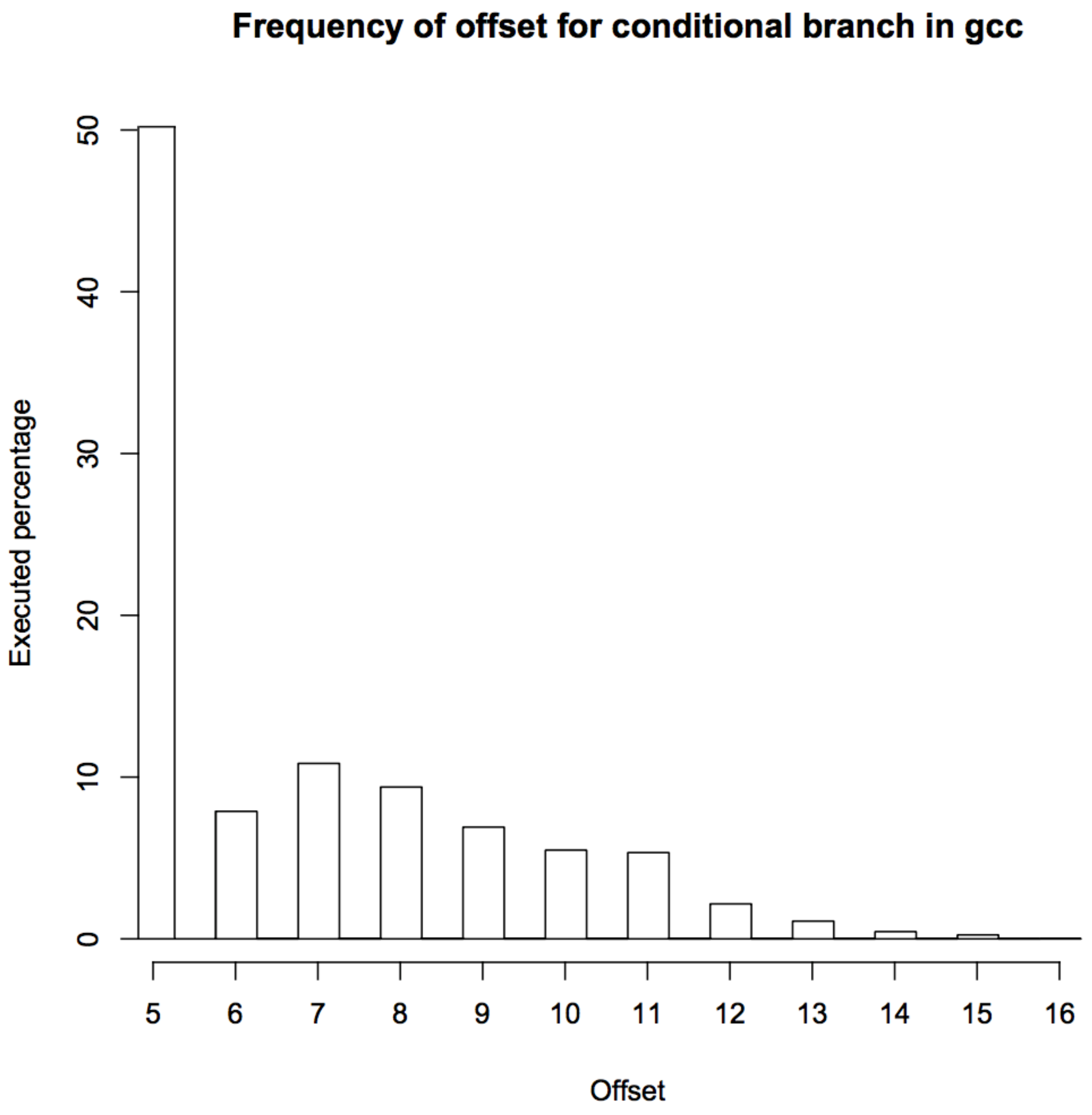
Jay Chang
39787114
Oct 5 2015

FIGURE 3.



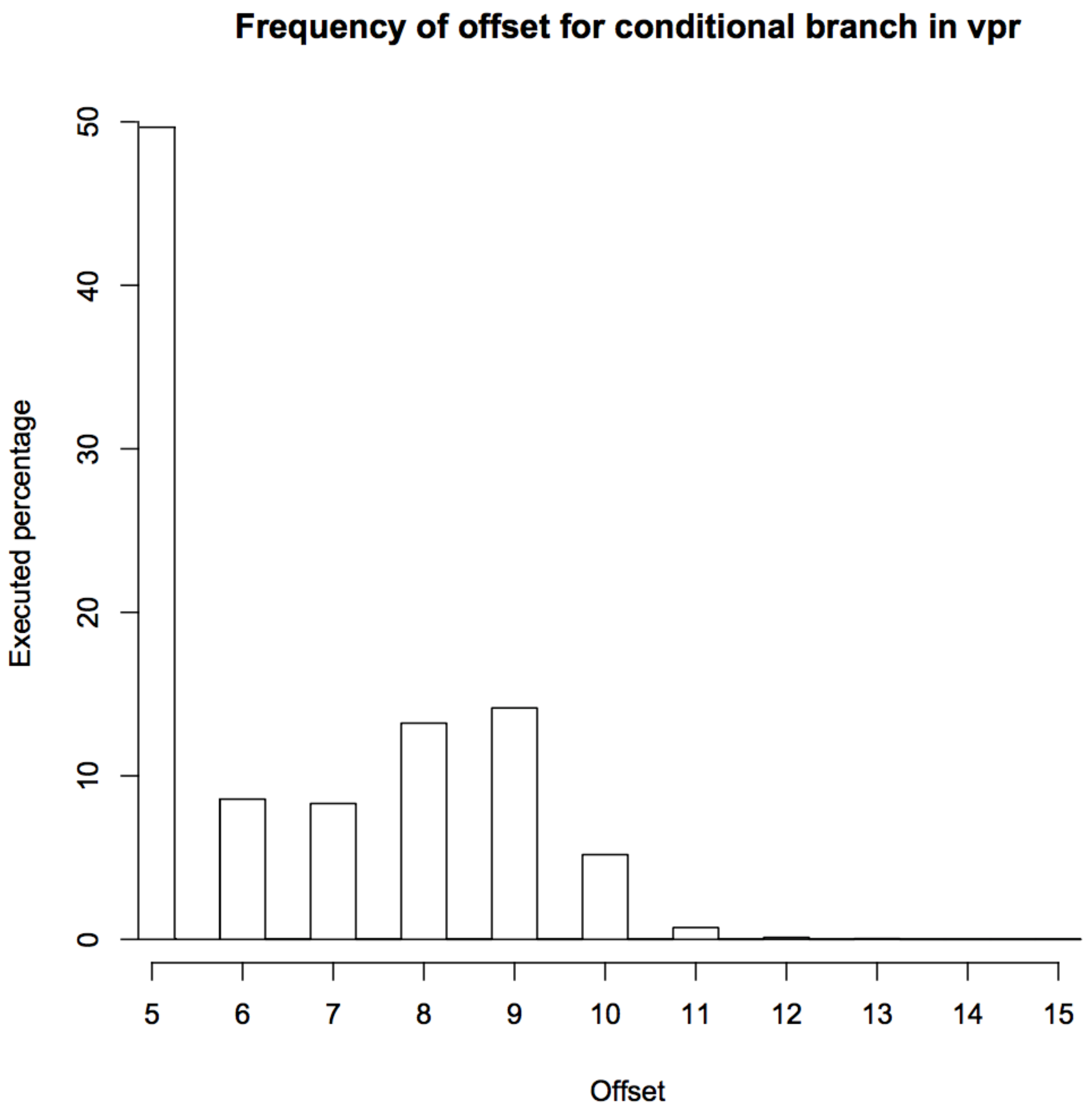
Jay Chang
39787114
Oct 5 2015

FIGURE 4.



Jay Chang
39787114
Oct 5 2015

FIGURE 5.



Jay Chang
39787114
Oct 5 2015

FIGURE 6.

The average number of bits that change when a general purpose register is written to	
fpppp	7.3624
go	6.7475
gcc	7.3105
vpr	6.5369