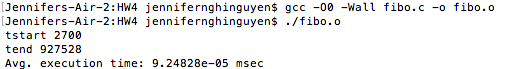
Nghi Nguyen

SJSU id: 010872316

HW4

**Question 3:**

(a) First, to get a baseline (without any explicitly declared registers), compile and run fibo.c.



(b) Now, modify the fibonacci() function by making the variables a, b, and c register variables. Recompile and run the code. How long does a single iteration take now, on average?

register unsigned int a = 0;

register unsigned int b = 1;

register unsigned int c;

int n;

/Users/jennifernghinguyen/Desktop/Screen Shot 2017-04-20 at 10.40.31 AM.png

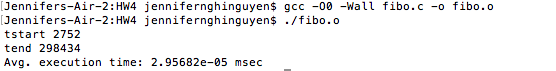
(c) Modify the fibonacci() function one more time by making the variable n also a register variable. Recompile and run the code once more. How long does a single iteration take with all four variables as register variables?

register unsigned int a = 0;

register unsigned int b = 1;

register unsigned int c;

register int n;



(d) Comment on your observed results. What can you conclude about using registers in your code?

In general, using registers in code will increase performance. In (b), using register to store variable a, b, c, the performance is almost 2 times faster; and in (c) store a, b, c and n using register, the performance increase by a factor of 3.

**Question 4**

**int extract\_n(int x, int pos) {**

**return (x >> pos) & 1;**

**}**

extract\_n: extract the bit at position nth

1. shift right x n position

2. then x and 1 to get the last bit

**int set\_n(int x, int pos) {**

**return x | (1 << pos);**

**}**

set\_n: insert 1 in nth position

1. shift 1 left n position

2. then, x or (shifted 1)

**int main() {**

**int x = 10;**

**int i = 32;**

**int y = 0,b;**

**while (--i >= 0 ) {**

**b = extract\_n(x, i);**

**if (b == 0)**

**y = set\_n(y, i);**

**printf("%d", b);**

**}**

**printf("\n%d\n", y);**

**}**

Result:

00000000000000000000000000001010

-11

main program prints binary representation of x = 10 (line 1) and y: decimal value of 1’s complement of x (or negation of x) (line 2)