Lab 1 Basic Debugging with Keil: Lab Report

Student: Jeremiah Webb

Student ID: 2545328

Instructor: Dr. Jianhua Liu

Section #2

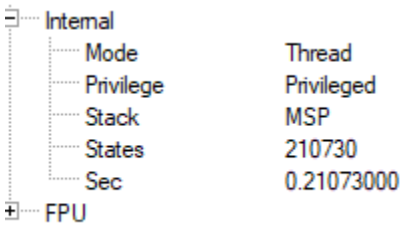
Introduction

This lab was used to analyze and understand more advanced ways to debug C programs and even compare assembly programmed functions to see the difference in speeds. Using the same algorithms in C and assembly we can see clock speed differences too. Using such performance evaluations can also help debug code and see how we can make programs overall faster and more efficient.

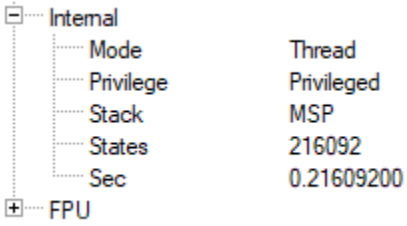
Tasks

Task 1

Beginning Clock Cycle:

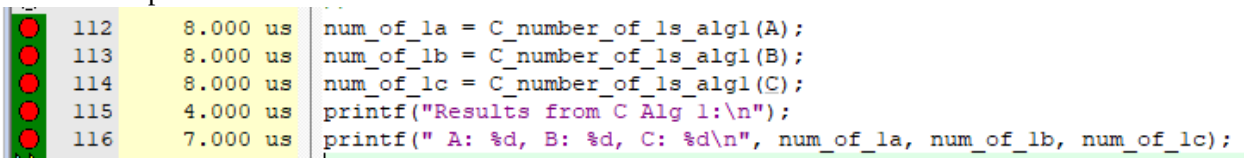


Ending Clock Cycles:



Complete Difference: 5362 clocks

Total Time spent: 35 us



Task 2

	B	C	D	E	F
		Clock Cycle Totals			
		Beginning Clock Cycles at 6284			
		Input A	Input B	Input C	
	Alg1, C	N/A	18	204	
	Alg1, asm	7057	16	202	
	Alg2, C	7128	23	215	
	Alg2, asm	7449	16	112	
	Alg3, C	7062	18	130	
	Alg3, asm	7353	16	128	
	Clock Beginnings				
	Alg1, C	6284	6302	6506	
	Alg2, C	13634	13657	13872	
	Alg3, C	20934	20952	21082	
	Alg1, asm	28139	28155	28357	
	Alg2, asm	35806	35822	35934	
	Alg3, asm	43287	43303	43431	

Lower Clock amounts means faster.

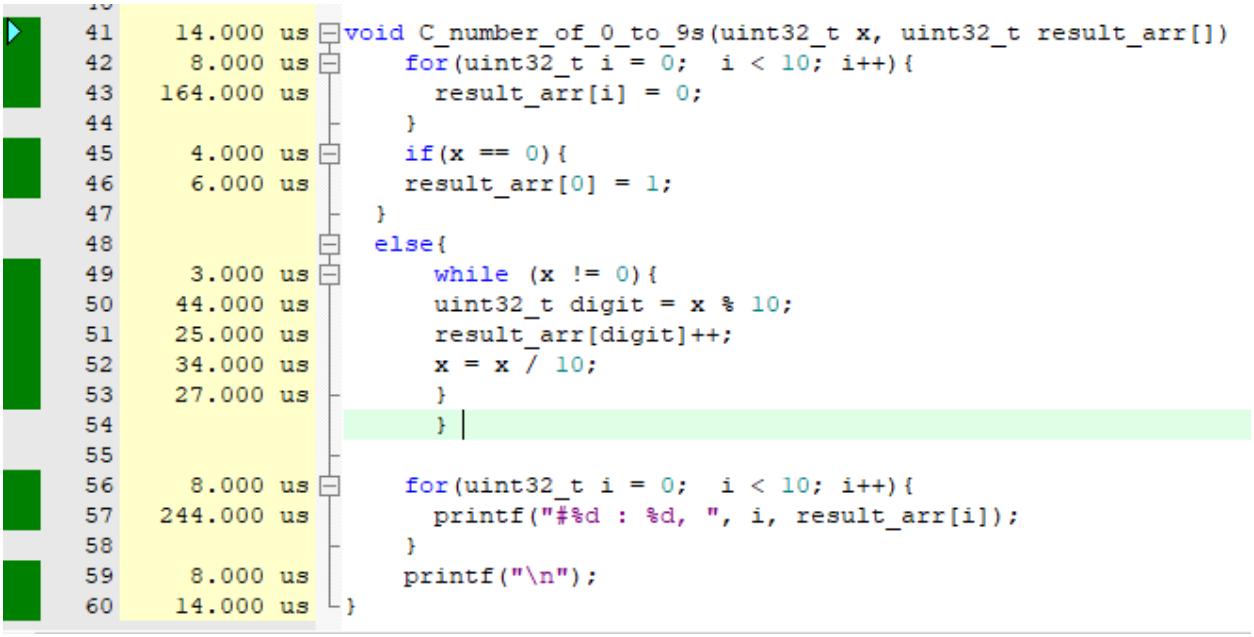
As shown, Assembly is significantly faster, for example between Alg2, C and Alg2,asm, Alg2,asm is 190% faster than its C counterpart. Overall, Assembly is faster, even in small algorithms.

Task 4

```
Counting for the decimal digits in 55320:#0 : 1, #1 : 0, #2 : 1, #3 : 1, #4 : 0, #5 : 2, #6 : 0, #7 : 0, #8 : 0, #9 : 0,

Counting for the decimal digits in 0:#0 : 1, #1 : 0, #2 : 0, #3 : 0, #4 : 0, #5 : 0, #6 : 0, #7 : 0, #8 : 0, #9 : 0,
```

Results



```
    14.000 us  void C_number_of_0_to_9s(uint32_t x, uint32_t result_arr[])
     8.000 us      for(uint32_t i = 0;  i < 10; i++){
   164.000 us        result_arr[i] = 0;
                   }
     4.000 us      if(x == 0){
     6.000 us      result_arr[0] = 1;
                  }
                  else{
     3.000 us        while (x != 0){
    44.000 us        uint32_t digit = x % 10;
    25.000 us        result_arr[digit]++;
    34.000 us        x = x / 10;
    27.000 us        }
                     }

     8.000 us      for(uint32_t i = 0;  i < 10; i++){
   244.000 us        printf("#%d : %d, ", i, result_arr[i]);
                   }
     8.000 us      printf("\n");
    14.000 us  }
```

Code

Code Snippets

Task 4

```
void C_number_of_0_to_9s(uint32_t x, uint32_t result_arr[]) {
          for(uint32_t i = 0;  i < 10; i++){
               result_arr[i] = 0;
          }
          if(x == 0){
          result_arr[0] = 1;
     }
     else{
               while (x != 0){
               uint32_t digit = x % 10;
               result_arr[digit]++;
               x = x / 10;
               }
               }
          for(uint32_t i = 0;  i < 10; i++){
               printf("#%d : %d, ", i, result_arr[i]);
          }
          printf("\n");
}
```

Narrative

The Lab did go well, seeing the difference between Assembly and C is a good insight into how our future labs when programming Assembly. Visualizing the speed difference is also a good insight into why programmers even program in Assembly, than a higher language like C.

Results

One can see that between C and Assembly, Assembly is significantly faster. In small algorithms one could argue that the difference is negligible, however in longer, more intricate functions and algorithms, Assembly is faster. Using clock speeds is additionally a great way to debug and understand how to make algorithms more efficient, in both languages.