**Project 2 Report**

Christopher Badolato

[badolato.christopher@knights.ucf.edu](mailto:badolato.christopher@knights.ucf.edu)

EEL 3801-0011: Computer Organization

3/18/2019

1. Project Description
2. Program Design
3. Symbol and Label Tables

|  |  |
| --- | --- |
| Register | Assignment |
| $v0 | Tells Mars when to Syscall. |
| Sa0 | Base address of our strings used throughout the program  add $a0, $a0, 1  takes us to the next of our string before we loop |
| $s0 | Stores the value of each string character as we loop through our string. |
| $t0-$t9 | Stores the integer count of each character “u, c ,f, k, n, i, g, h, t, s” respectively (s is stored in $s0). |
| $s1 | Used as our repeated loop counter throughout the printing of the hashtag histogram. |

Each loop Label has a corresponding exit condition.

|  |  |
| --- | --- |
| Label | Use |
| loopThroughString:  exit: | Represents a while loop, which loops through our string. (although in the c code prototype I used a for loop and string length). We add to our base address of the string to continue this loop to the next character.  We will exit this loop only if our value stored within our loaded bit register is null. |
| nextChar1: - nextChar10: | Within the above loop we essentially create if else statements  If is equal, add 1 to our count at that character, check the next character. |
| nextLoop1: - nextLoop10 | Keeps continuous loops going until our branch breaks us to our next character loop. |
| nextPrintLoop1: - nextPrintLoop10 | Branches to our next characters hashtag printing loop if our loop counter hasn’t reached the total character count. |

1. Learning Coverage

* Looping through a string in MIPS using a base address and register to store the current character.
* We can check if our character is null or 0 to find the end of our string.
* The ability to compare characters of a string to their ascii value to separate out certain characters or in our case, take a count.
* Utilizing loops to create and print a histogram.

1. Prototype in C

Code can be easily copy and pasted into IDE.

|  |
| --- |
| // Christopher Badolato 3064088  // 3/18/2019  // EEL3801 0011  /\* this code will conduct the given functions from project 2.  It represents the MIPS code we are writing, which involves creating a histogram  using only the letts from "UCF KNIGHTS" the string entered must be LOWERCASE it does  not .  \*/  #include <string.h>  #include <stdio.h>  int main(){  //created integer variables to represent the count at each letters  //as well as grabs our string length, This code only works if entered string  //is all lowercase  int loopCounter = 0, strLen, charAsciiValue;  int u = 0, c = 0, f = 0, k = 0, n = 0 ,i = 0 ,g = 0, h = 0, t = 0,s = 0;  char\* string = "the knights field 16 varsity teams (6 men's, 9 women's and one co-ed sport) that have won numerous national and conference titles. the ucf knights football team won conference championships in 2007 and 2010, and the knights women's basketball team won conference titles the 2009 and 2010.";  strLen = strlen(string);  //Checks each char in the string to see if it is part of our UCFKNIGHTS char list  // we will add 1 to the count if the "see" one of our selected characters.  for(loopCounter = 0; loopCounter < strLen; loopCounter++){  charAsciiValue = (int)string[loopCounter];  //u  if(charAsciiValue == 117){  u++;  }  //c  else if(charAsciiValue == 99){  c++;  }  //f  else if(charAsciiValue == 102){  f++;  }  //k  else if(charAsciiValue == 107){  k++;  }  //n  else if(charAsciiValue == 110){  n++;  }  //i  else if(charAsciiValue == 105){  i++;  }  //g  else if(charAsciiValue == 103){  g++;  }  //h  else if(charAsciiValue == 104){  h++;  }  //t  else if(charAsciiValue == 116){  t++;  }  //s  else if(charAsciiValue == 115){  s++;  }  //Go to the next character in the string once we have checked  //each character we'd like to make the histogram for. "ucfknights"  else{  continue;  }  }  //prints our integer histogram.  printf("u: %d\n", u);  printf("c: %d\n", c);  printf("f: %d\n", f);  printf("k: %d\n", k);  printf("n: %d\n", n);  printf("i: %d\n", i);  printf("g: %d\n", g);  printf("h: %d\n", h);  printf("t: %d\n", t);  printf("s: %d\n", s);  printf("\n");  //Based on the values from the histogram above we will print pounds or  //"hashtags" for each character count by looping through that character count.  //u  printf("u: ");  for(loopCounter = 0; loopCounter < u; loopCounter++){  printf("#");  }  printf("\n");  //c  printf("c: ");  for(loopCounter = 0; loopCounter < c; loopCounter++){  printf("#");  }  printf("\n");  //f  printf("f: ");  for(loopCounter = 0; loopCounter < f; loopCounter++){  printf("#");  }  printf("\n");  //k  printf("k: ");  for(loopCounter = 0; loopCounter < k; loopCounter++){  printf("#");  }  printf("\n");  //n  printf("n: ");  for(loopCounter = 0; loopCounter < n; loopCounter++){  printf("#");  }  printf("\n");  //i  printf("i: ");  for(loopCounter = 0; loopCounter < i; loopCounter++){  printf("#");  }  printf("\n");  //g  printf("g: ");  for(loopCounter = 0; loopCounter < g; loopCounter++){  printf("#");  }  printf("\n");  //h  printf("h: ");  for(loopCounter = 0; loopCounter < h; loopCounter++){  printf("#");  }  printf("\n");  //t  printf("t: ");  for(loopCounter = 0; loopCounter < t; loopCounter++){  printf("#");  }  printf("\n");  //s  printf("s: ");  for(loopCounter = 0; loopCounter < s; loopCounter++){  printf("#");  }  return 0;  } |

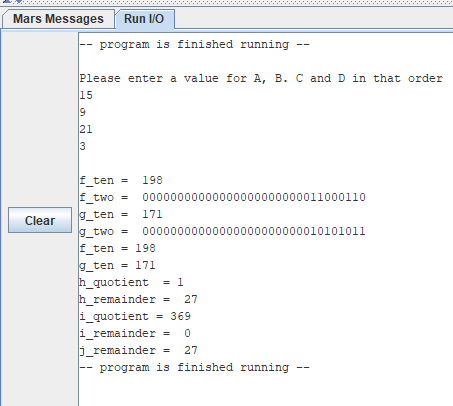
1. Test Plan

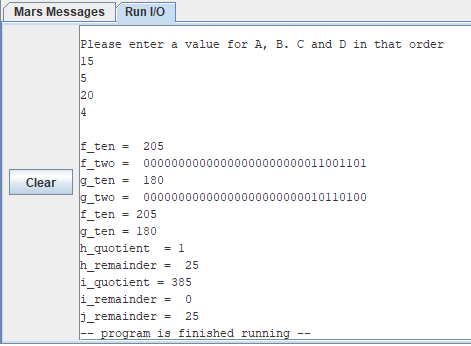
To test our Mips code, we have applied various test cases to ensure our output is correct. Each case corresponds to the results shown below in the images. We have tested and verified with our C code that these outputs correspond to the output of our Mips run version

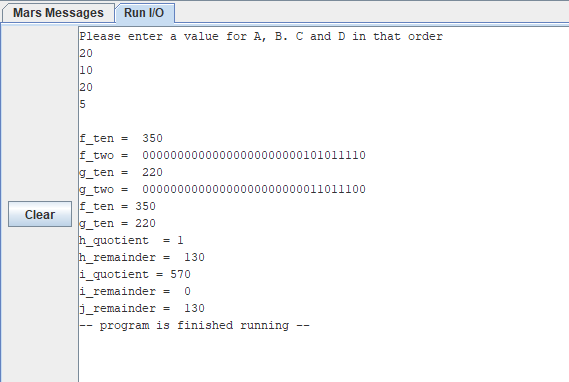
|  |  |
| --- | --- |
| Inputs | Outputs |
| A = 15  B = 9  C = 21  D = 3 | F = 198  G = 171  h\_quotient = 1  h\_remainder = 27  i\_quotient = 369  i\_remainder = 0  j\_remainder = 27 |
| A = 15  B = 5  C = 20  D = 4 | F = 236  G = 186  h\_quotient = 1  h\_remainder = 52  i\_quotient = 420  i\_remainder = 0  j\_remainder = 52 |
| A = 20  B = 10  C = 20  D = 5 | F = 350  G = 220  h\_quotient = 1  h\_remainder = 130  i\_quotient = 570  i\_remainder = 0  j\_remainder = 130 |

1. Test Results

The results of my test cases are below. I was having some issues with numbers that would change the values to all 0’s. Any instance where we would end up with an undefined fraction, or a case such that G > F, would result in the program crashing when finding i\_remainder. So, I just tested it with a few cases. Testing each of these in our C code also resulted in the below outputs.







1. References

* EEL 3801 00219 DeMara Slides, Module 03-MIPS-ISA.pdf.
* Mars 4.5 Mips Simulator
* Recitation files
* Testing reference Sheet