CS3843 Programming Assignment #4: Assembly Coding with Arrays (70 points)

**©2019 Larry W. Clark, this document must not be copied to any other website**

**In this program, you must code the following three functions using IA32 on a hen server:**

int **count** (pszStr, cMatch)

int **countAny3**(pszStr, pszC3)

int **substr**(pszStr, iBeg, iEnd, pszResult)

Notes:

1. Your code must be executed on a hen server.
2. For assembling and linking your code, use the **make** utility. I provided the **makefile.**

**make p4**

1. Executing your program, use**:**

**./p4 <p4Input.txt**

1. To help limit your creative use of the gcc compiler to generate the assembly language, I have provided skeletal code for **each of the functions.** You must use the registers in the way they are described in the header comments. If you use gcc to generate the assembly instructions for any of these functions, you will receive a zero on this assignment. Note that you will be doing IA32 assembly coding on the final exam
2. You **must** use **pointer advancement**.
3. Recall that you shouldn't do any other statements (including calling **dumpRegs**, **hexDumpEbp**, **hexDumpEsp**), immediately after a comparison since this will cause a loss of the condition code flag values.

**int count (pszStr, cMatch)** (10 pts)

Complete the IA32 assembly code for the **count** function which is passed the address of a string array of characters (zero terminated) and a match character (not a string). **count** returns a count of the number of occurrences of the match character in the string.

Example: count("ASTROS", 'S') returns 2

**int countAny3 (pszStr, pszC3)** (20 pts)

Complete the IA32 assembly code for the **countAny3** function which is passed the address of a string array of characters (zero terminated) and a string of three match characters. **countAny3** returns a count of the number of occurrences of any of the three characters in the string. You may assume that the second parameter has three characters (not including the zero byte).

Example: countAny3("HAPPYDAYS", "PYS") returns 5



**int substr(pszStr, iBeg, iEnd, pszResult)** (40 pts)

Complete the IA32 assembly code for the **substr** function which is passed the address of a string array of characters (zero terminated), a beginning subscript, an ending subscript, and the address of a resulting array of characters (should be zero terminated by you).

* It returns the substring of the string beginning with iBeg and ending with one position before iEnd.
* It also functionally returns the number of characters in the result string (not including the zero byte).
* If iEnd is past the end of the string, simply stop the string at the end. This is not an error.
* If iBeg < 0, set the result to an empty string (i.e., set its first character to 0x00), and return 0.
* if iBeg >= iEnd, set the result to an empty string and return 0.

Examples: Assume szSource is "mickey"

a. **substr**(szSource, 1, 4, szResult) would return "ick" via the 4th parm and functionally return 3.

b. **substr**(szSource, 1, 20, szResult) would return "ickey" via the 4th parm and functionally return 5.

c. **substr**(szSource, -5, 4, szResult) would return "" via the 4th parm and functionally return 0.

d. **substr**(szSource, 4, 4, szResult) would return "" via the 4th parm and functionally return 0.

**Byte Movement and Comparison (%dl is just as example register)**

* Use **movb *address,* %dl** to move a byte at the specified address to the one byte register %dl. The *address* is probably in a register.
* Use **movb %dl, *address*** to move a byte from the one byte register %dl to the specified address which is probably in a register.
* Use **cmpb *value*, %dl** to compare a value with the one byte register %dl.

**Bonus (10 pts + 150 points / n )**

1. **char \*verify(pszStr, pszValid)**

Complete the IA32 assembly code for the **verify** function which is passed the address of a string array and the address of an array of valid characters. You must check to see whether each character in the string array appears somewhere in the array of valid characters. If all of the characters in the string are valid, return NULL (i.e., 0x00000000). Otherwise, return the address of the first character that doesn't match.

Examples:

a. **verify**("catch", "abcdhijstr") would return NULL since each character of "catch" is valid.

b. **verify**("donkey", "abcdhijstr") would return the address of the "o" in pszStr.



1. If your code on all of the functions is correct according to all requirements, well documented, and **not late**, you will receive 10 points + 150 points / n where n is the number of people who meet all requirements.

**I have provided the following files in /usr/local/courses/clark/cs3843/Pgm4Hen :**

1. **cs3843p4Driver.o** - the driver which invokes count, countAny3, substr, and verify.
2. **count.s** - a bare minimum skeleton for **count**. You will be coding and documenting this function. After you copy this to your directory, you might need to chmod +w
3. **countAny3.s** - a bare minimum skeleton for **countAny3**. You will be coding and documenting this function. After you copy this to your directory, you might need to chmod +w
4. **substr.s** - a bare minimum skeleton for **substr**. You will be coding and documenting this function. After you copy this to your directory, you might need to chmod +w
5. **verify.s** - a bare minimum skeleton for **verify**. You will be coding and documenting this function for extra credit. After you copy this to your directory, you might need to chmod +w
6. **p4Input.txt** – Used by the driver to invoke the functions.
7. **dumpRegs.o** - dumpRegs writes the contents of the 8 major registers to stdout. It does not receive any parameters.
8. **hexDump.o** - a hexDump function (as described in assignment #1). It also prints the address being dumped.
9. **hexDumpEbp.o** – calls **hexDump** passing %ebp, 84, 12. This function preserves all registers and restores them; therefore, you can use it without any preparation except as noted above.
10. **hexDumpEsp.o** – calls **hexDump** passing %esp, 84, 12. This function preserves all registers and restores them; therefore, you can use it without any preparation except as noted above.
11. **makefile** – uses the **make** utility to compile whichever code changes and links them

**Turn In:**

1. Your fully coded and documented code for **count.s**, **countAny3.s**, and **substr.s**. Please refer to the Assembler documentation standards which is on my website under Programming Assignments.
2. If you do the bonus, include your code for **verify.s**.
3. Your output which should not call hexDump, hexDumpEbp, hexDumpEsp nor hexDump.