

CSE 220 Homework Assignment #1

File Edit View Tools Help

Request edit access

Share

these labels. Rather, the starting addresses of the strings are stored at those labels. You will need to use load instructions to obtain the contents of these strings stored at the addresses: `lw` to load the address of a string, then multiple `lbu` instructions to get the characters. For instance, `lw $s0, addr_arg2`, followed by `lbu $t0, 4($s0)` would store the character located at index 4 in `$t0` (assuming 0-based indexes) of the third command line argument.

Running the Program

Running the provided `hwk1.asm` file is pretty simple. Hit F3 on the keyboard or press the button shown below to assemble your code:



If your code has any syntax errors, MARS will report them in the MARS **Messages** panel at the bottom of the window. Fix any syntax errors you may have. Then press F5 or hit the Run button shown below to run your program:



Any output generated by your program will appear in the **Run I/O** panel at the bottom of the window.

Part 1: Validate the First Command-line Argument and the Number of Command-line Arguments

For this assignment you will be implementing several operations that perform computations and do data manipulation.

In `hwk1.asm`, begin writing your program immediately after the label called `start_coding_here`. You may declare more items in the `.data` section after the provided code. Any code that has already been provided must appear exactly as defined in the given file. Do not delete or rename these labels, as doing so will negatively impact grading.

The number of arguments is stored in memory at the label `num_args` by code already provided for you. You will need to use various system calls to print values that your code generates. For example, to print a string in MIPS, you need to use system call 4. You can find a listing of all the official MARS

Screenshot of a Google Docs page titled "CSE 220 Homework Assignment #1". The page contains instructions for a MIPS program assignment.

The number of arguments is stored in memory at the label `num_args` by code already provided for you. You will need to use various system calls to print values that your code generates. For example, to print a string in MIPS, you need to use system call 4. You can find a listing of all the official MARS system calls on the [MARS website](#). You can also find the documentation for all instructions and supported system calls within MARS itself. Click the blue question mark in the right-hand end of the tool bar to open it.

Later in the document is a list of the operations that your program will execute. Each operation is identified by a single character. The character is given by the first command-line argument (whose address is `addr_arg0`). The parameter(s) to each argument are given as the remaining

4

command-line arguments (located at addresses `addr_arg1`, `addr_arg2`, etc.). In this first part of the assignment your program must make sure that each operation is valid and has been given the correct number of parameters. Perform the validations in the following order:

1. The first command-line argument must be a string of length one that consists of one of the following characters: 1, 2, S, F, R or P. If the argument is a letter, it must be given in uppercase. If the argument is not one of these strings or if the argument contains more than one character, print the string found at label `invalid_operation_error` and exit the program (via system call 10). This string contains a newline character at the end, so you do not need to provide your own.
2. The 1, 2 and S operations expect two additional arguments. If the total number of command-line arguments for these commands is not three, print the string found at label `invalid_args_error` and exit the program (via system call 10). This string contains a newline character at the end, so you do not need to provide your own.
3. The F operation expects one additional argument. If the total number of command-line arguments for this command is not two, print the string found at label `invalid_args_error` and exit the program (via system call 10). This string contains a newline character at the end, so you do not need to provide your own.

docs.google.com

CSE 220 (44 unread) | ASCII Code - The extended ASCII table | MIPS syscall functions available in MA... | CSE220 | Department of Computer Sc... | CSE 220 - Weekly Schedule | CSE 220 Homework Assignment #1 - ... +

CSE 220 Homework Assignment #1

File Edit View Tools Help Request edit access Share

4. The R operation expects six additional arguments. If the total number of command-line arguments for this command is not seven, print the string found at label `invalid_args_error` and exit the program (via system call 10). This string contains a newline character at the end, so you do not need to provide your own.

5. The P operation expects one additional argument. If the total number of command-line arguments for this command is not two, print the string found at label `invalid_args_error` and exit the program (via system call 10). This string contains a newline character at the end, so you do not need to provide your own.

Important: You must use the provided `invalid_operation_error` and `invalid_args_error` strings when printing error messages. Do not create your own labels for printing output to the screen. If your output is marked as incorrect by the grading scripts because of typos, then it is your fault for not using the provided strings, and you will lose all credit for those test cases.

Be sure to initialize all of your values (e.g., registers) within your functions. Never assume registers or memory will hold any particular values (e.g., zero). MARS initializes all of the registers and bytes of main memory to zeroes. The grading scripts for later assignments will fill the registers and/or main memory with random values before executing your code.

Later sections will explain how to validate the arguments for each operation.

Examples of Invalid Input

5

See the later sections of the documents for explanation of what arguments are valid for each operation.

| Command-line Arguments | Label for String to Print | Expected Output |
|------------------------|---------------------------|-----------------|
|------------------------|---------------------------|-----------------|

docs.google.com

CSE 220 (44 unread) | ASCII Code - The extended ASCII table | MIPS syscall functions available in MA... | CSE220 | Department of Computer Sc... | CSE 220 - Weekly Schedule | CSE 220 Homework Assignment #1 - ... +

CSE 220 Homework Assignment #1

File Edit View Tools Help Request edit access Share

| Command-line Arguments | Label for String to Print | Expected Output |
|------------------------|---------------------------|-------------------|
| Signed 000E 32 | invalid_operation_error | INVALID_OPERATION |
| p HIJKL | invalid_operation_error | INVALID_OPERATION |
| F XYZ ABC | invalid_args | INVALID_ARGS |

Character Strings in MIPS Assembly

In assembly, a string is a one-dimensional array of unsigned bytes. Therefore, to read each character of the string we typically need to use a loop. Suppose that `$s0` contains the base address of the string (that is, the address of the first character of the string). We could use the instruction `lbu $t0, 0($s0)` to copy the first character of the string into `$t0`. To get the next character of the string, we have two options: (i) add 1 to the contents of `$s0` and then execute `lbu $t0, 0($s0)` again, or (ii) leave the contents of `$s0` alone and execute `lbu $t0, 1($s0)`. Generally speaking, the first approach is easier and simpler to use, but the other approach works perfectly fine if the string is short in length and is always the same length. Note that syntax like `lbu $t0, $t1($s0)` is not valid; an immediate value (a constant) must be given outside the parentheses.

Next: Process the Input

If the program determines that the first command-line argument is a valid operation and that it has been given a correct number of additional arguments, the program continues by executing the appropriate operation as specified below. Note that you are permitted to add code to the `.data` section as necessary to implement these operations.

Part 2: Process a String of Four Hexadecimal Digits that Represents a Signed Integer

The 1, 2 and S operations treat the second command-line argument as a string of hexadecimal digit characters ('0' – '9', 'A' – 'F') that represent a signed one's complement number, two's complement number or sign/magnitude number, respectively. The leftmost character represents the most significant nibble (4 bits) of the integer. The operation converts the string into a 32-bit signed integer in two's complement, storing it in a single register. The code then prints the N rightmost bits of that register, where N is given by the third argument. In other words, the integer is printed as an N -bit, two's complement value. Finally, the code prints a newline character ('\n'). The number of bits to print given by the third argument should be at least 16. Helpful hint: you can give an ASCII character

docs.google.com

CSE 220 (44 unread) | ASCII Code - The extended ASCII table | MIPS syscall functions available in MA... | CSE220 | Department of Computer Sc... | CSE 220 - Weekly Schedule | CSE 220 Homework Assignment #1 - ... +

CSE 220 Homework Assignment #1

File Edit View Tools Help Request edit access Share

In two's complement, storing it in a single register. The code then prints the N rightmost bits of that register, where N is given by the third argument. In other words, the integer is printed as an N -bit, two's complement value. Finally, the code prints a newline character ('\n'). The number of bits to print given by the third argument should be at least 16. Helpful hint: you can give an ASCII character as a literal in MIPS (e.g., li \$a0, '\n').

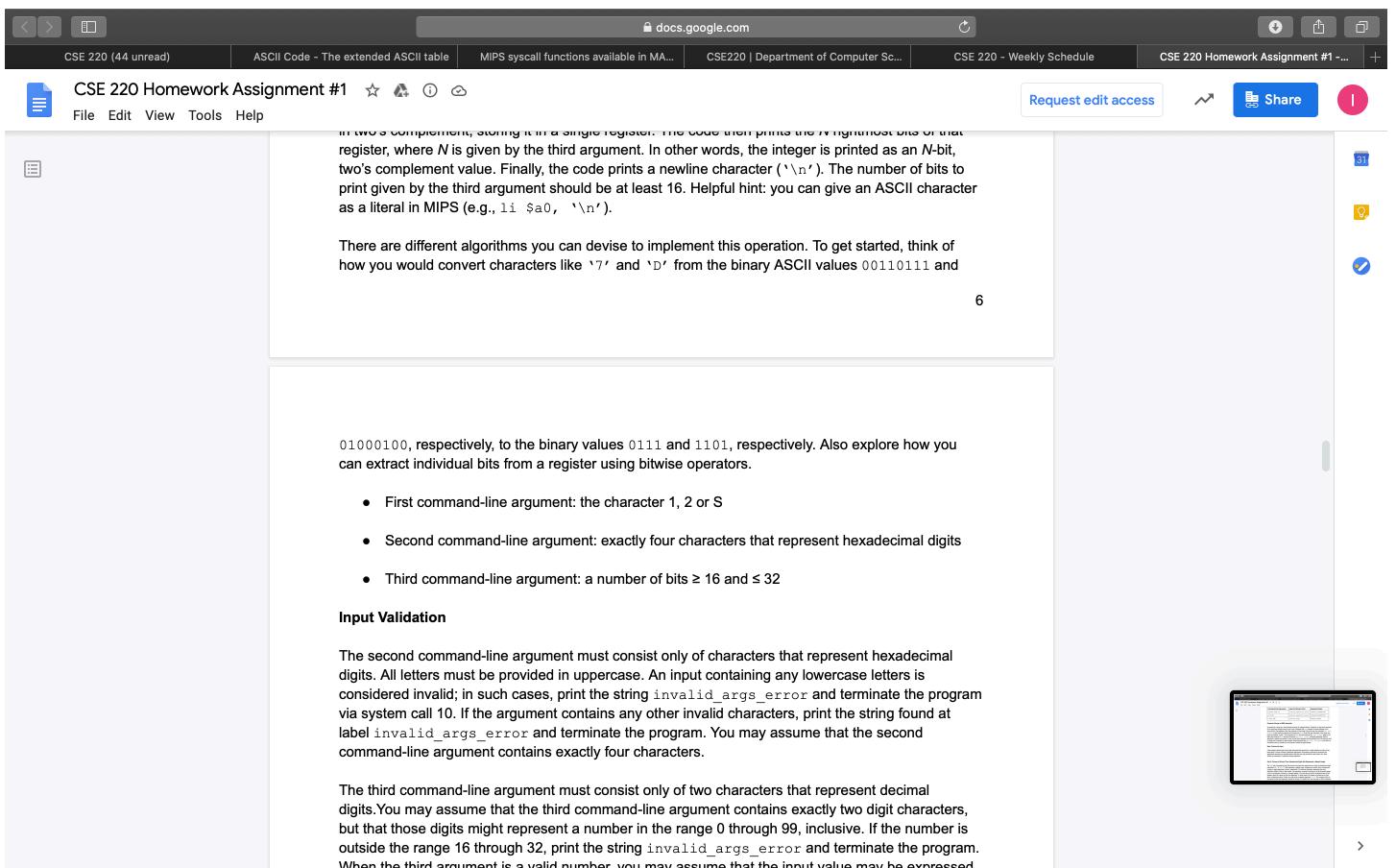
There are different algorithms you can devise to implement this operation. To get started, think of how you would convert characters like '7' and 'D' from the binary ASCII values 00110111 and 01000100, respectively, to the binary values 0111 and 1101, respectively. Also explore how you can extract individual bits from a register using bitwise operators.

- First command-line argument: the character 1, 2 or S
- Second command-line argument: exactly four characters that represent hexadecimal digits
- Third command-line argument: a number of bits ≥ 16 and ≤ 32

Input Validation

The second command-line argument must consist only of characters that represent hexadecimal digits. All letters must be provided in uppercase. An input containing any lowercase letters is considered invalid; in such cases, print the string `invalid_args_error` and terminate the program via system call 10. If the argument contains any other invalid characters, print the string found at label `invalid_args_error` and terminate the program. You may assume that the second command-line argument contains exactly four characters.

The third command-line argument must consist only of two characters that represent decimal digits. You may assume that the third command-line argument contains exactly two digit characters, but that those digits might represent a number in the range 0 through 99, inclusive. If the number is outside the range 16 through 32, print the string `invalid_args_error` and terminate the program. When the third argument is a valid number, you may assume that the input value may be expressed



docs.google.com

CSE 220 (44 unread) | ASCII Code - The extended ASCII table | MIPS syscall functions available in MA... | CSE220 | Department of Computer Sc... | CSE 220 - Weekly Schedule | CSE 220 Homework Assignment #1 - ... +

CSE 220 Homework Assignment #1

File Edit View Tools Help Request edit access Share

Worked-out Example

Suppose the command-line arguments are "1" "FFF1" "25". These arguments indicate that the string "FFF1" represents a one's complement value and that we want to print out the two's complement representation of that value using 25 bits.

Note that $\text{FFF1}_{16} = 1111\ 1111\ 1111\ 0001_2$. The first argument of "1" indicates that we should treat this bitstring as the one's complement value of some integer. $1111\ 1111\ 1111\ 0001$ represents the integer -14 in one's complement representation, which is represented in 25-bit two's complement as $1111111111111111111110010$, which is the output of the code.

Examples

| Command-line Arguments | Expected Output |
|------------------------|---------------------------|
| 2 FFFF 20 | 1111111111111111111111111 |
| 2 0011 25 | 0000000000000000000010001 |
| 2 7EA2 19 | 000011111010100010 |

7

| | |
|-----------|--------------------------------|
| 1 FFFE 20 | 1111111111111111111111111 |
| 1 FFF1 25 | 11111111111111111111111010 |
| 1 4190 32 | 000000000000000010000110010000 |
| S 000E 22 | 000000000000000000001110 |
| S 800A 20 | 111111111111111111110110 |
| S 00FF 19 | 00000000000111111111 |

Screenshot of a Google Docs page titled "CSE 220 Homework Assignment #1". The page contains a table showing command-line arguments and their expected binary outputs, followed by instructions for Part 3 and examples.

| Command-line Arguments | Expected Output |
|------------------------|--------------------|
| S 00FF 19 | 000000000011111111 |
| 2 faCE 19 | INVALID_ARGS |
| 1 WOLF 28 | INVALID_ARGS |
| S 01F1 05 | INVALID_ARGS |

Part 3: Print a Decimal Fixed-point Number as a Binary Fixed-point Number

The F operation takes a string of nine characters that represent a positive decimal real number and, using only integer arithmetic, converts the decimal value to binary and prints it to the screen. The whole number part of the output must not contain any leading zeros except in the case where the whole number part is zero. In this case, only a single zero may be printed. Finally, the code prints a newline character ('\n').

The fractional part of the input number will always represent a value that can be computed by adding some combination of the values 0.5, 0.25, 0.125, 0.0625 and 0.03125 (that is, 2^{-1} , 2^{-2} , 2^{-3} , 2^{-4} and 2^{-5}). (This accommodation makes it possible to convert the fractional decimal value to binary using only integer arithmetic.) The output value must always contain exactly five bits to the right of the radix point.

- First command-line argument: the character F
- Second command-line argument: exactly three decimal digits characters, followed by a period, followed by exactly five decimal digit characters

Input Validation

You may assume that the input is always valid.

Examples

| Command-line Arguments | Expected Output |
|------------------------|-----------------|
|------------------------|-----------------|

docs.google.com

CSE 220 (44 unread) | ASCII Code - The extended ASCII table | MIPS syscall functions available in MA... | CSE220 | Department of Computer Sc... | CSE 220 - Weekly Schedule | CSE 220 Homework Assignment #1 - ... +

CSE 220 Homework Assignment #1

File Edit View Tools Help Request edit access Share

| | |
|-------------|------------------|
| F 612.68750 | 1001100100.10110 |
| F 006.40625 | 110.01101 |
| F 000.62500 | 0.10100 |
| F 000.03125 | 0.00001 |
| F 017.00000 | 10001.00000 |
| F 195.18750 | 11000011.00110 |

Part 4: Encode Six Numerical Fields as an R-Type MIPS Instruction

The R operation takes six decimal values, treats them as the six fields of a MIPS R-type instruction, and, using shifting and masking operations, combines the four values into a single 32-bit integer that represents an R-type instruction. The program then prints this value in hexadecimal using syscall 34. Finally, the code prints a newline character ('\\n'). An example will help to clarify this process.

- First command-line argument: the character R
- Second command-line argument: the string "00"
- Third command-line argument: a string that encodes a positive, two-digit decimal integer in the range [0, 31] (the rs field). A leading zero is provided if needed.
- Fourth command-line argument: a string that encodes a positive, two-digit decimal integer in the range [0, 31] (the rt field). A leading zero is provided if needed.
- Fifth command-line argument: a string that encodes a positive, two-digit decimal integer in the range [0, 31] (the rd field). A leading zero is provided if needed.
- Sixth command-line argument: a string that encodes a positive, two-digit decimal integer in the range [0, 31] (the shamt field). A leading zero is provided if needed.
- Seventh command-line argument: a string that encodes a positive, two-digit decimal integer in the range [0, 63] (the funct field). A leading zero is provided if needed.

A 32 bit R-type instruction consists of 6 fields, where the number below each field indicates its width

docs.google.com

CSE 220 (44 unread) | ASCII Code - The extended ASCII table | MIPS syscall functions available in MA... | CSE220 | Department of Computer Sc... | CSE 220 - Weekly Schedule | CSE 220 Homework Assignment #1 - ... +

CSE 220 Homework Assignment #1

Request edit access Share

A 32-bit R-type instruction consists of 6 fields, where the number below each field indicates its width in bits. Note that bit #0 is the rightmost bit:

| Field | opcode | rs | rt | rd | shamt | funct |
|-----------|--------|----|----|----|-------|-------|
| # of bits | 6 | 5 | 5 | 5 | 5 | 6 |

9

Suppose the command-line arguments were the strings

R 00 15 17 03 14 29

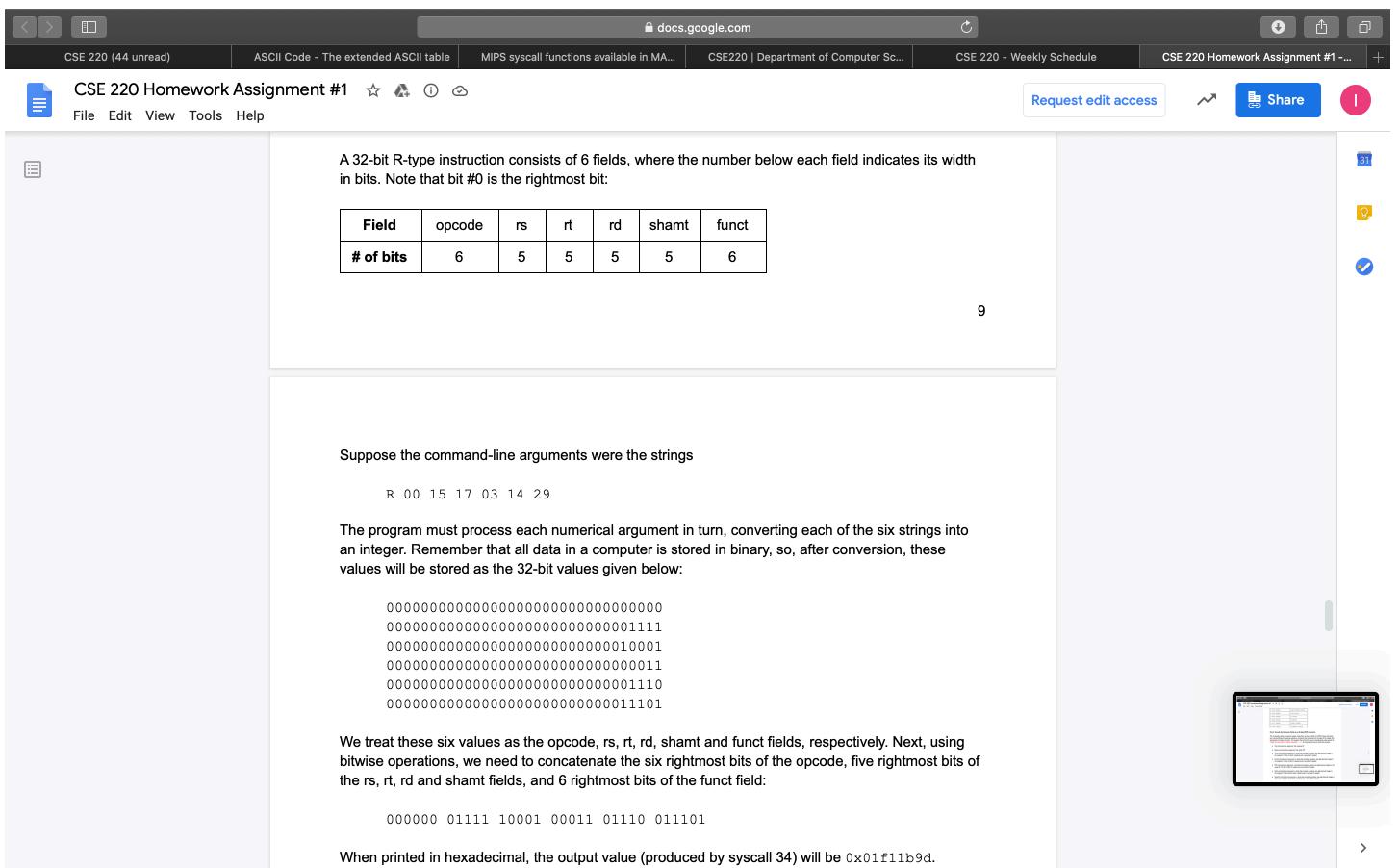
The program must process each numerical argument in turn, converting each of the six strings into an integer. Remember that all data in a computer is stored in binary, so, after conversion, these values will be stored as the 32-bit values given below:

```
00000000000000000000000000000000
000000000000000000000000000000001111
0000000000000000000000000000000010001
000000000000000000000000000000000001
000000000000000000000000000000001110
0000000000000000000000000000000011101
```

We treat these six values as the opcode, rs, rt, rd, shamt and funct fields, respectively. Next, using bitwise operations, we need to concatenate the six rightmost bits of the opcode, five rightmost bits of the rs, rt, rd and shamt fields, and 6 rightmost bits of the funct field:

```
000000 01111 10001 00011 01110 011101
```

When printed in hexadecimal, the output value (produced by syscall 34) will be 0x01f11b9d.



docs.google.com

CSE 220 (44 unread) | ASCII Code - The extended ASCII table | MIPS syscall functions available in MA... | CSE220 | Department of Computer Sc... | CSE 220 - Weekly Schedule | CSE 220 Homework Assignment #1 - ... +

CSE 220 Homework Assignment #1

Request edit access Share

File Edit View Tools Help

Input Validation

You may assume that the second argument is always "00". You may assume that the third, fourth, fifth, six and seventh command-line arguments always consist of exactly (and only) two decimal digit characters that encode a positive integer. However, you may not assume that the values are in the legal ranges specified above. If any of the third, fourth, fifth, six and seventh command-line arguments is outside its legal range, print the string found at label `invalid_args_error` and exit the program (via system call 10).

Note that the shamt field is applicable only when performing a shift instruction. You do not need to check whether a non-zero shamt amount is given when a non-shifting instruction is provided. Simply have your code convert the shamt argument to a 5-bit binary value, regardless of what the funct field is, and incorporate that 5-bit value in the final 32-bit value computed by your code.

Examples

| Command-line Arguments | Expected Output |
|------------------------|-----------------|
| R 00 15 17 03 14 29 | 0x01f11b9d |
| R 00 02 31 24 05 42 | 0x005fc16a |
| R 00 22 25 31 00 61 | 0x02d9f83d |

10

R 00 17 41 22 30 08 | INVALID_ARGS

Part 5: Identify Non-standard, Five-card Poker Hands

The P operation treats the second command-line argument as a string of exactly five characters that encode a [non-standard five-card hand](#) from draw poker.

docs.google.com

CSE 220 (44 unread) | ASCII Code - The extended ASCII table | MIPS syscall functions available in MA... | CSE220 | Department of Computer Sc... | CSE 220 - Weekly Schedule | CSE 220 Homework Assignment #1 - ... +

CSE 220 Homework Assignment #1

File Edit View Tools Help Request edit access Share

Part 5: Identify Non-standard, Five-card Poker Hands

The P operation treats the second command-line argument as a string of exactly five characters that encode a [non-standard five-card hand](#) from draw poker.

The program must be able to identify some of the non-standard hands of five-card draw poker and print a message indicating which one it found. If the hand can be matched with two or more hands, the **hand** of highest rank is printed. If none of these five ranked hands is identified, the program prints HIGH_CARD. You may assume that when the P operation has been provided, the second argument always contains exactly 5 characters. For the purposes of this assignment, an Ace is always treated as a low card (i.e., less than 2), never as a high card (i.e., higher than King).

The hand ranks from highest-to-lowest, along with the relevant MIPS string to print, are:

| Rank | Hand Name | Label for String to Print | Output on Screen |
|------|-----------------------------------|---------------------------|------------------|
| 1 | Big bobtail | big_bobtail_str | BIG_BOBTAIL |
| 2 | Full house | full_house_str | FULL_HOUSE |
| 3 | Five and dime | five_and_dime_str | FIVE_AND_DIME |
| 4 | Skeet | skeet_str | SKEET |
| 5 | Blaze | blaze_str | BLAZE |
| 6 | Everything else (i.e., High card) | high_card_str | HIGH_CARD |

From Wikipedia:

1. Big bobtail: A four-card straight flush (four cards of the same suit in consecutive order).
2. Full house: three cards of one rank and two cards of another rank (example: 3-3-3-J-J).
3. Five and dime: 5-low, 10-high, with no pair (example: 5-6-7-8-10).
4. Skeet: A hand with a 2, a 5, and a 9, plus two other un-paired cards lower than 9 (example: 2-4-5-6-9).
5. Blaze: All cards are Jacks, Queens, and/or Kings.
6. High card: None of the above.

A single card is encoded as an ASCII character. Specifically, the ASCII code ranges employed are:

- 0x41 - 0x4D represent the A through K of Clubs
- 0x51 - 0x5D represent the A through K of Spades

docs.google.com

CSE 220 (44 unread) | ASCII Code - The extended ASCII table | MIPS syscall functions available in MA... | CSE220 | Department of Computer Sc... | CSE 220 - Weekly Schedule | CSE 220 Homework Assignment #1 - ... | +

CSE 220 Homework Assignment #1

File Edit View Tools Help Request edit access Share

Note that the left hexadecimal digit encodes the suit, and the right digit encodes the rank. Valid ranks range 1 - 13 (0x1 - 0xD). Note that 11, 12 and 13 (0xB, 0xC, 0xD) correspond with Jack, Queen and King, respectively.

For example, 0x63 represents the 3 of Diamonds and corresponds with the ASCII character 'c'. Likewise, 0x4B represents the Jack of Clubs and corresponds with the ASCII character 'K'.

An example of a valid hand would be the string "vX[qd", which have ASCII codes:

- 0x76 = 6 of Hearts
- 0x58 = 8 of Spades
- 0x5B = Jack of Spades
- 0x71 = Ace of Hearts
- 0x64 = 4 of Diamonds

Input Validation

You may assume that the second command-line argument always consists of exactly five characters that encode valid cards. You may also assume that no card appears more than once in the input.

Examples

| Command-line Arguments | Meaning of Input | Expected Output |
|------------------------|------------------|-----------------|
| P BVTUS | 2♦6♣4♦5♦3♦ | BIG_BOBTAIL |
| P BDCEG | 2♦4♦3♦5♦7♦ | BIG_BOBTAIL |
| P RTtbB | 2♦4♦4♥2♦2♦ | FULL_HOUSE |
| P WUxjI | 7♦5♦8♥T♦9♦ | FIVE_AND_DIME |
| P WRuiF | 7♦2♦5♦9♦6♦ | SKEET |
| P TRSYU | 4♦2♦3♦9♦5♦ | BIG_BOBTAIL |
| P [L]mK | J♦Q♦K♥K♦J♦ | BLAZE |
| P GIDuF | 7♦9♦4♦5♦6♦ | HIGH_CARD |