Welcome to the Lab

CMPUT 229

University of Alberta

Fall 2022

Outline

Lab Requirements

- Assembly control flow
- Loading and storing from memory
- Using syscalls
- Function symantics

Luhn's Algorithm: Commonly used Algorithm for generation of valid CreditCard numbers

- Is there any math even behind it? or just random numbers of specified length?
- Created by IBM scientist Hans Peter Luhn to validate a variety of identification numbers such as credit card number, IMEI numbers, Social Insurance Number (SINs), etc...

Luhn's Algorithm: Commonly used Algorithm for generation of valid CreditCard numbers

- This relation isn't two ways. Not every valid Luhn's number is a credit card number as different companies follow different patterns to generate credit card number on the top of Luhn's check.
- For a credit card number to be valid in practice, credit card companies needs to issue and register it.
- Examples:
- 4539999147413350 - Valid Luhn's Number
- 3845062241375445 - Invalid Luhn's Number

Modulo Operator

- A brief refresher on modulo operator, which is used to find remainder of division:
  - $10 \% 2 = 0$
  - $10 \% 3 = 1$
- In RISC-V assembly, the **rem** operator is equivalent to the modulo operator % in other languages.
- In RISC-V, rem t0, t1, t2 stores t1 % t2 into t0.

CreditCard Validation

- Your assignment will require to implement a valid luhn's algorithm checker in RISC-V assembly.
- Luhn's algorithm is as follows.
  - Step 1: Starting from the rightmost digit, double the value of every second digit.
  - Step 2: If doubling a number in Step 1 results in a two digit number, add the digits of the product to get a single digit number. For example, doubling the number 9 ($9 \times 2 = 18$) results in a two digit number 18, so we will add the digits ($1 + 8 = 9$) to get 9.

CreditCard Validation

- Continued Algorithm:
  - Step 3: Add all the digits in the array. For example, if the array is [1, 2, 3, 4] then the sum is $1 + 2 + 3 + 4 = 10$.
  - Step 4: If the sum is divisible by 10, then the number is valid. In other words, if the sum ends in 0 or the remainder (modulo) by 10 is 0, then the number is valid.

Credit Card Classification

- Classification of Valid Credit Card numbers:
  - MasterCard cards begin with the numbers 51, 52 or 54 and have 16 digits.
  - Diner's Club cards begin with the numbers 300, 301, 302, 303, 304, and 305, and have 14 digits.
  - VISA cards begin with the number 4 and have 13 or 16 digits. Furthermore, if a VISA number starts with 455953, it is a VISA Chase card.

Assignment

- Write a function called *creditValid* that classifies if a given credit card is valid using Luhn's algorithm, and saves modified array in memory reserved by a2.
- Input:
  - *a0*: pointer to an array of digits(32-bit integers) of number expression
  - *a1*: length of array
  - *a2*: pointer to space reserved to store modified array of digits
- Return:
  - *a0*: type of card
    - 0 if card is invalid
    - 1 if card is VISA but not issued by chase, 12 if issued by chase
    - 2 if card is MasterCard
    - 3 if card is Diner's Club
    - 4 if card is unknown

System Calls

- A list of system calls (syscalls) supported by RARS can be found in the RARS help page.
- The syscall you may use in this lab is PrintInt, which prints the integer stored in the a0 register to standard output.
- The PrintInt syscall is executed once you set a7 (call number) to 1 followed by ecall instruction.
- Example Usage:
  ```
  li a0, 2
  li a7, 1
  ecall
  ```
- The above set of instructions print 2 to standard output.

Assignment Tips

- Read specifications very carefully. Pay special attention to what you have to include - we don't want a `main:` label.
- Test your assignments on the lab machines before you submit. That's where we'll be marking them.
- Look at the marksheet to get an idea of how the grading will be done.
- Style marks are easy marks. Format your code like the `example.s` file we provided, and write good comments.
- Be sure to submit code that runs and compiles.
- Every function in RISC-V needs a return statement. At the end of your function's execution, return with the instruction 'jr ra' or pseudo instruction 'ret'.
- Note: in this lab, do not print any newlines in order to ensure that the grading scripts understand your solution.

Debugging your lab

- RARS has a builtin debugger which lets you add breakpoints pseudoinstruction ebreak or from graphical interface.
- Setting a breakpoint will automatically stop the execution of the code before the next instruction, allowing you to inspect the state of your program.
- Using syscalls to print on screen is also helpful for debugging.

Questions?