Lab 2

CMPUT 229

University of Alberta

Outline

Lab 2: ASCII and Tables

Hexadecimal: Why?

- Hexadecimal is frequently used due to the easy translation to and from binary.
- It's a lot easier to read and represent HEX than binary.
- Also: translation between hexadecimal and ASCII is easy, as well as from ASCII to hexadecimal.

ASCII

- American Standard Code for Information Interchange
- In short: a representation standard for characters and symbols.
- How to convert from ASCII numbers to binary?
  - ASCII numbers are encoded between 0x30 and 0x39.
  - Subtracting the offset leaves the number in binary representation.
  - Each number has to be added according to it position in the string array (i.e. units, tens, hundreds, etc).

Tables in Assembly

- Tables in assembly are simple: it's a base address and a certain amount of allocated space.
- **Why?** $\leftarrow$ makes dealing with specific memory needs easier.
- It is helpful when using data chunks larger than one word (to prevent misalignment).

Subroutines

- *Subroutines* in RISC-V are calls to another portion of the code.
- **Why?** ← Just like "procedures" or "functions" in higher level languages.
- `jal` is an instruction to make the jump to a symbol (marker for a piece of code).
- `ret` return control to the caller.
- Conventions: a registers for arguments, a0/a1 for return values.
- This will be examined more in detail in next labs.

The Assignment

You are required to code four subroutines:

- *readHex* - read an hexadecimal ASCII number (0-9, A-F) and return the number if valid.
- *printHex* - read an integer value and print the hexadecimal ASCII representation (0-9,A-F).
- *createCountTable* - create a table to track 200 integers and a counter for each.
- *countIntegerAccess* - receives an integer and returns the number of times accessed (including this one).

readHex

- *Parameter:* a0 contains the memory address of a non-null terminated, 8 character string.
  - e.g. “ab00123d” or even ”string01”
- *Return:*
  - a1 = {0,1} a boolean to determine if the string is valid or not (1 = invalid).
  - a0 = the number represented by the string in binary.

printHex

- *Parameter:* a0 contains a valid integer value.
- *Output:* Prints on the string using ecalls the hexadecimal ASCCI representation, e.g. 0x001255AF. Do not place a newline at the end.

createCountTable

- This function does not receive arguments, but it should initialize the table (integer + counter) with enough room for 200 integers.
- Counters *must* be initialized to zero.
- A flag must be used to initialize integer spaces too.

countIntegerAccess

- *Parameter:* a0 an arbitrary integer value.
- *Return:*
  - a0 = the number of times this function has been called with this integer.

Important: the table must be updated each for the specific number each time it is called, and if it is not in the table, insert it.

Assignment Tips

- Read specifications very carefully. Pay special attention to what you have to include - we don't want a `main` method.
- You can use the "test.s" file to test your subroutine implementation.
- 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 `test.s` file we provided, and write good comments.
- Be sure to submit code that runs and compiles. Otherwise you will lose many marks.

Questions?