

CMPUT 229

## Lab #3: BCD Multiplication

CMPUT 229

# Background

# What is Binary Coded Decimal (BCD)?

## **Alternative encoding system:**

Instead of representing numbers in binary, represent each decimal digit separately.

Each decimal digit is represented in 4 bits.

BCD encoded digits are identical to the binary encoding but are truncated to 4 bits.

0	0000	5	0101
1	0001	6	0110
2	0010	7	0111
3	0011	8	1000
4	0100	9	1001

# Why do we care about BCD?

## **Arbitrarily large numbers:**

A processor can only perform arithmetic on certain builtin types.

32-bit RISC-V only supports 32-bit integers and 32-bit floats .

We can represent arbitrarily large numbers precisely with BCD.

## **Control over precision and rounding:**

Floating point rounding can cause many issues.

Financial and business applications require strict rounding.

# Variable Length BCD Format

## **Why is this format needed?**

A single BCD digit can only store the numbers from 0 to 9.

If we want to store larger numbers we can use multiple digits.

The digits must be stored in some standardized format.

## **Description of the format:**

Numbers are encoded as arrays of bytes, with two BCD digits per byte.

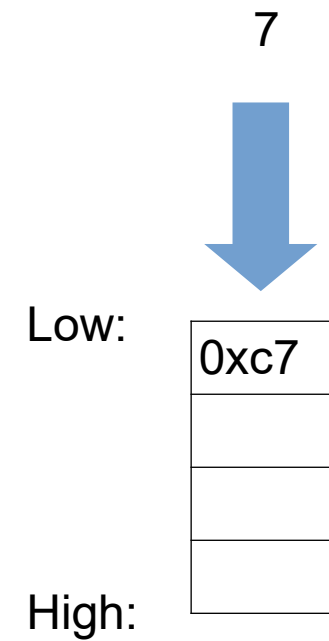
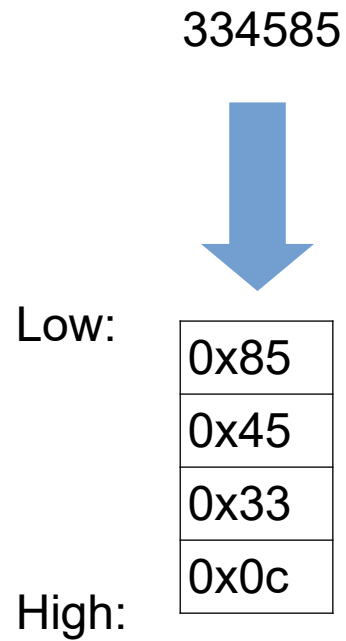
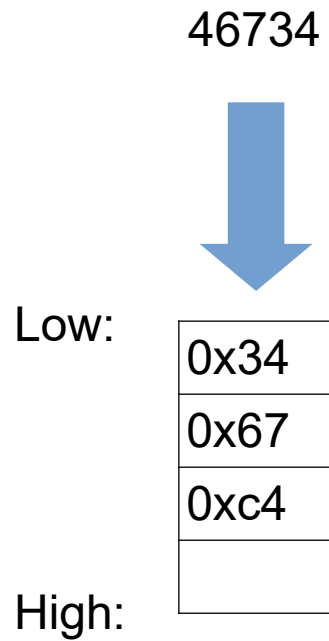
The digit with the lower place value is stored in the lower nibble of the byte, while the digit with the higher place value is stored in the higher nibble.

The lowest byte contains the lowest two digits, and so on.

The number is terminated with the value 0xc, stored one nibble higher than the last digit.

Numbers contain no leading 0's, except of the number 0, which is encoded as 0xc0.

## BCD Format Examples



# Addition algorithm

## Informal description:

Assume input numbers A and B

Sum each digit pair, starting from the lowest pair.

If a number becomes larger than 10, split the number by place value and set the current result to the digit with the lower place value. Then propagate the digit with the higher place value so it is included in the next addition.

$$\begin{array}{r} 3\ 9\ 0\ 2 \\ +\ 4\ 5\ 8\ 5 \\ \hline \end{array}$$

$$\begin{array}{r} 3\ 9\ 0\ 2 \\ +\ 4\ 5\ 8\ 5 \\ \hline 7 \end{array}$$

$$\begin{array}{r} 3\ 9\ 0\ 2 \\ +\ 4\ 5\ 8\ 5 \\ \hline 8\ 7 \end{array}$$

$$\begin{array}{r} 1 \\ 3\ 9\ 0\ 2 \\ +\ 4\ 5\ 8\ 5 \\ \hline 4\ 8\ 7 \end{array}$$

$$\begin{array}{r} 1 \\ 3\ 9\ 0\ 2 \\ +\ 4\ 5\ 8\ 5 \\ \hline 8\ 4\ 8\ 7 \end{array}$$

# Multiplication algorithm

## **Informal description:**

Assume input numbers A and B.

For each digit of B, multiply that digit by each digit of A.

Shift the partial product by the place value of the digit of B. As in, the first product should not be shifted, the should be by 1 to the left and so on.

Add the shifted partial products.

## **Hints:**

You may find the RISC-V multiplication, division, and remainder instructions useful.



# Multiplication Example

$$\begin{array}{r} 9429 \\ \times 385 \\ \hline \end{array}$$

$$\begin{array}{r} 4 \\ 942\boxed{9} \\ \times 38\boxed{5} \\ \hline 5 \end{array}$$

$$\begin{array}{r} 14 \\ 94\boxed{2}9 \\ \times 38\boxed{5} \\ \hline 45 \end{array}$$

$$\begin{array}{r} 214 \\ 9\boxed{4}29 \\ \times 38\boxed{5} \\ \hline 145 \end{array}$$

$$\begin{array}{r} 4214 \\ \boxed{9}429 \\ \times 38\boxed{5} \\ \hline 7145 \end{array}$$

$$\begin{array}{r} 4214 \\ 9429 \\ \times 38\boxed{5} \\ \hline 47145 \end{array}$$

$$\begin{array}{r} 9429 \\ \times 3\boxed{8}5 \\ \hline 47145 \\ 75432 \end{array}$$

$$\begin{array}{r} 9429 \\ \times \boxed{3}85 \\ \hline 47145 \\ 75432 \\ 28287 \end{array}$$

$$\begin{array}{r} 9429 \\ \times 385 \\ \hline 47145 \\ 75432 \\ 28287 \\ \hline 3630165 \end{array}$$

CMPUT 229

# Helper Functions

# Helper Functions

## Included Functions

We have included some functions to operate on binary BCD numbers.

These functions can get and set a particular digit of a binary BCD number, as well as compute the length of a binary BCD number.

# getNth

**Description:**

Get the nth digit of a binary BCD number. The lowest digit has index 0.

**Parameters:**

- a0: Pointer to a binary BCD number.
- a1: Index of the digit to get.

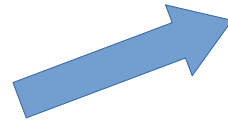
**Return Value:**

- a0: nth digit

# getNth Example

Suppose getNth is called with the following inputs:

a0	0x10001000
a1	1



0x45	0x23	0xc1	
------	------	------	--

Number is 12345

After the call, the result in a0 will be 4, since 4 is the 2<sup>nd</sup> digit.

# setNth

**Description:**

Set the nth digit of a binary BCD number. The lowest digit has index 0.

**Parameters:**

a0: Pointer to a binary BCD number.

a1: Index of the digit to set.

a2: Digit to set the given index to

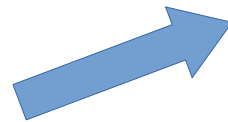
**Return Value:**

N/A

# setNth Example

Suppose setNth is called with the following inputs:

a0	0x10001000
a1	4
a2	9



0x45	0x23	0xc1	
------	------	------	--

Number is 12345

After the call, the BCD number at address 0x10001000 will be modified:

0x45	0x23	0xc9	
------	------	------	--

Number is 92345

# length

**Description:**

Get the length of a binary BCD number, including the terminating character.

**Parameters:**

a0: Pointer to a binary BCD number.

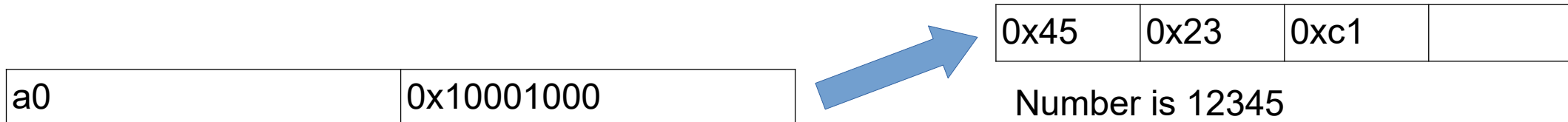
**Return Value:**

a0: Length of the input number



# length Example

Suppose length is called with the following inputs:



After the call, `a0` will have the value 6. This is because the input number has 5 digits, and the terminator is counted.

CMPUT 229

# Assignment

# Overview of the lab

## **Overview:**

In this lab, you must implement addition and multiplication of BCD numbers, using the given format.

Additionally, you must write small functions to read from a string to the BCD format, and to print a BCD number.

# mulBCD

**Description:**

This function takes two BCD numbers A and B, and prints their product. Internally, the multiplication must be performed on binary BCD numbers.

**Parameters:**

a0: Pointer to an ASCII encoded integer (A).

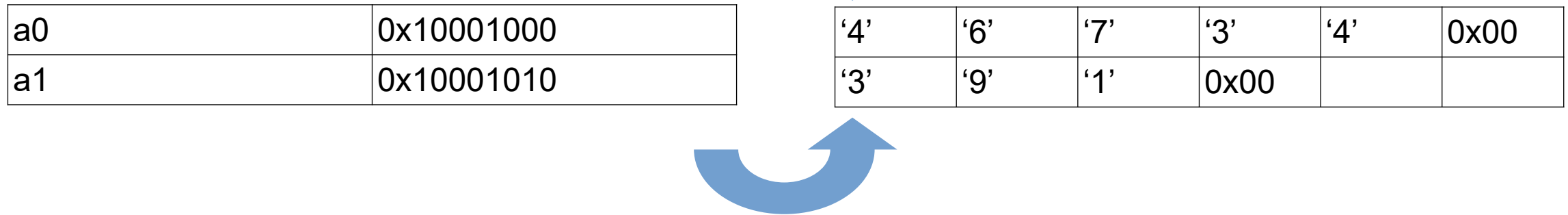
a1: Pointer to an ASCII encoded integer (B).

**Return Value:**

N/A

# mulBCD Example

Suppose mulBCD is called with the following inputs, with '1' representing the ASCII character of "1":



mulBCD should print the product of 46734 and 391, which is 18272994.

# addBCD

**Description:**

This function adds two BCD numbers and writes the result to a buffer. The sum is stored as a binary BCD number.

**Parameters:**

a0: Pointer to the first operand in binary BCD format.

a1: Pointer to the second operand in binary BCD format.

a2: Pointer to the result buffer

**Return Value:**

N/A

# readBCD

**Description:**

This function converts an input string into a BCD formatted number.

**Parameters:**

a0: Pointer to an ASCII encoded integer.

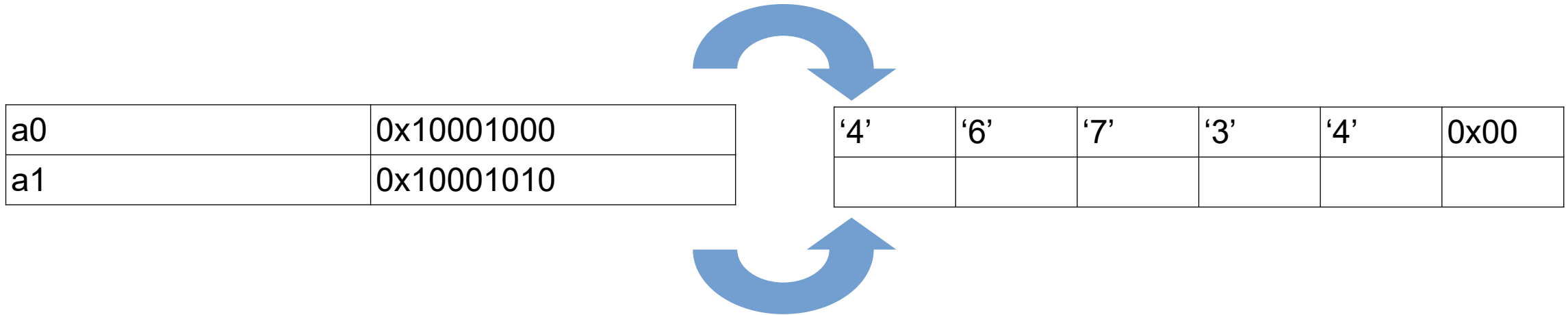
a1: Pointer to the result buffer.

**Return Value:**

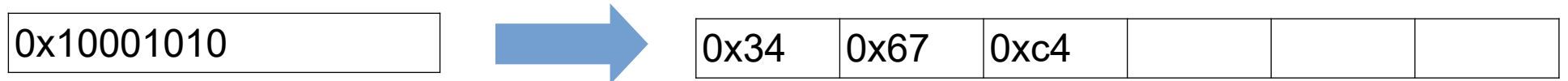
N/A

# readBCD Example

Suppose readBCD is called with the following inputs:



After the call to readBCD, the memory location 0x10001010 should contain the following:





# printBCD

**Description:**

This function prints a BCD number to standard output. A newline should be printed after the number.

**Parameter:**

a0: Pointer to an integer in binary BCD format.

**Return Value:**

N/A

# What to submit?

Submit a single file **multiplyBCD.s**.

Make sure to push your changes before the assignment deadline.

CMPUT 229

# Testing

# Testing your Solution

## Included tests:

We have provided some test inputs and outputs.

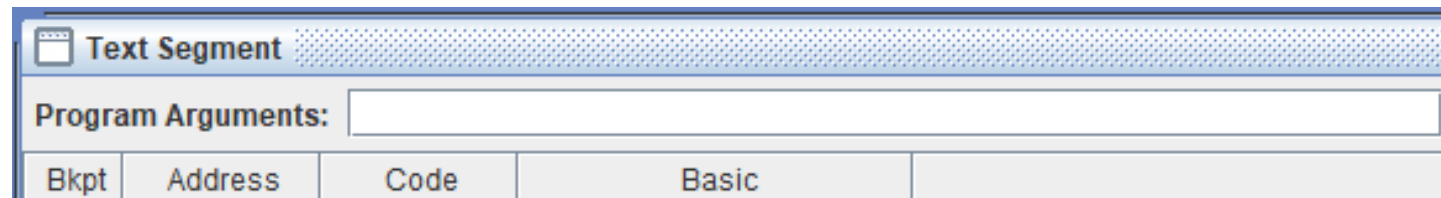
Tests are stored in the “Tests” directory. \*.txt files correspond to the inputs, while \*.out files are the expected outputs.

## Give file path as program arguments:

Give the path to the \*.txt file in the program arguments.

Make sure the test file has the correct format (Next slide).

RARS may need the full path to the test file.



Bkpt	Address	Code	Basic	

# Format of the Test File

## **A test file contains an input to mulBCD:**

The file must have two lines, each with a number.

The numbers in a test file must be valid inputs as given by the specification.