

# BigInteger Library in MIPS-32 Assembly

CS321/CS322 Mini-Project

Mukuntha N S, 1601CS27

# Introduction

---

- Primitive integer data type - 32-bits or 64-bits in C, C++, Java, etc.
- Calculations involving very large integers cannot be directly handled
- Eg.: Factorial of 100, which contains 158 characters
- Several programming languages big-integer libraries to handle such big integer operations (Java - BigInteger, C++ - Boost MultiPrecision, python - 'bignum').
- In this project, several basic functions have been implemented in the MIPS-32 instruction set.

# Description

---

- The big-integer 'struct':  
First 4-bytes stores size in bytes of the integer array.  
The rest is an integer array of a given size. The least-significant-bit is stored first (little-endian).
- This code currently supports a big integer of a maximum size of  $2^{32}$  bytes. All operations work on arbitrary length big integers.
- The code currently only supports unsigned operations, which can easily be extended to support signed big-integers if needed.

# Functions implemented - Utilities

— — —

- `make_bi`
- `make_bi_100`
- `ascii_to_int`
- `make_bi_from_str`
- `add_3_words`
- `print_bi`
- `set_bi_zero`

# Functions implemented - Utilities

— — —

- `add_bi_bi`
- `sub_bi_bi`
- `mult_bi_bi`
- `comp_bi_bi`

# add\_bi\_bi

---

This is a function that adds 2 big integers. Unsigned addition is done word-by-word. Returns \$v0 - address of new big integer. The function first makes a new big integer to store the result of size  $\max(l1, l2) + 4$  bytes where  $l1$ ,  $l2$  are the sizes in bytes of the input big integers. The addition is done word-by-word, and the carry is stored. The function calls the `add_3_words` subroutine for adding the two words and the carry word.

# sub\_bi\_bi

— — —

This is a function that finds the difference of 2 big-integers. Unsigned subtraction is done byte-by-byte. The smaller big-integer is subtracted from the smaller big-integer. Returns \$v0 - address of new big-integer. The comparison function is used to determine if the big integers need to be swapped. The function then makes a new big integer to store the result of size  $\max(l1, l2)$  bytes where  $l1, l2$  are the sizes in bytes of the input big integers. The subtraction is done byte-by-byte, and the borrow is stored if needed. When doing the subtraction, the borrow is added to the subtrahend, and then this sum is subtracted from the minuend ( $\text{minuend} - (\text{subtrahend} + \text{borrow}) = \text{difference}$ ). This is done byte-by-byte since the sum of the subtrahend and the borrow might not fit in a 32-bit register if done word-by-word.

# mult\_bi\_bi

---

This is a function that adds 2 big integers. Unsigned multiplication is done word-by-word. Returns \$v0 - address of new big integer.

The outer loop loops through the first big integer. The inner loop loops through the second. A new big integer of size  $l1 + l2$  is made and initialized to 0. Each word from the first big integer gets multiplied to the whole second big integer, with the carry handled. Again, the `add_3_words` subroutine is used to add the carry, the accumulated sum in the result, and the product of the words. This is then put in the right position in the result.



# comp\_bi\_bi

---

This is a function that compares two big integers. It returns boolean flags for 'equal', 'less-than', 'greater-than', 'less-than-or-equal', and 'greater-than-or-equal'.

Values passed - \$a0 - first bi, \$a1 - second bi.

Returns: \$v0 - Equal, \$a0 - LT, \$a1 - GT, \$a2 - LTE, \$a3 - GTE.

First, the lengths of the two big integers are compared. If they're equal, the integers are compared word-by-word, and the flags are set depending on the comparison.

Mars Messages

Run I/O

Clear

```
A is:
0x00011111 0x11111111
B is:
0x00000002 0x22222222 0x22222222
A == B is False
A < B is True
A > B is False
A <= B is True
A >= B is False
Sum is:
0x00000000 0x00000002 0x22233333 0x33333333
Difference is:
0x00000002 0x22211111 0x11111111
Product is:
0x00000000 0x0002468a 0xcf13579b 0xbbbb9753 0x0eca8642

-- program is finished running (dropped off bottom) --
```

Output showing comparisons, sum, difference, and product

**Thank you!**