Munin: A Simple Emulator for Execution-Time Memory Complexity Analysis

Fall 2023 CS 335 Final Project

Eleftheria Beres

Dec 2023

Hello

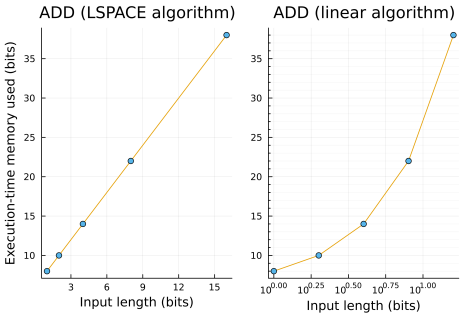

Figure 1: Execution-time memory usage (bits) vs. input length (bits) for the linear space ADD algorithm given in Appendix **??**

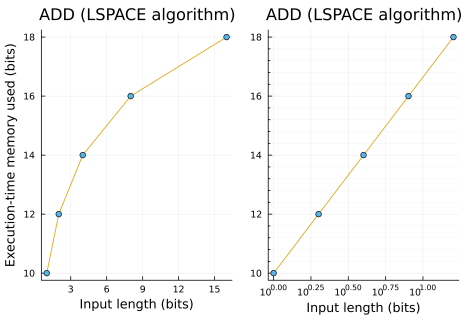

Figure 2: Execution-time memory usage (bits) vs. input length (bits) for the LSPACE ADD algorithm given in Appendix **??**

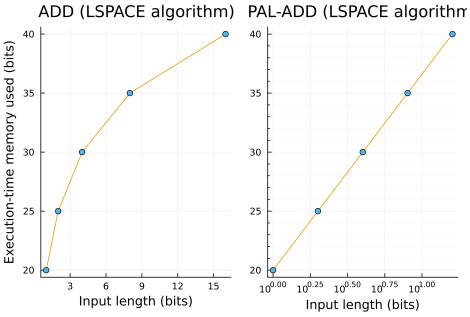

Figure 3: Execution-time memory usage (bits) vs. input length (bits) for the LSPACE PAL-ADD algorithm given in Appendix **??**

A Munin example algorithms

A.1 Linear-space ADD algorithm

A.1.1 Pseudo-code

```
function lin_add(x, y, z)
    a = x
    b = y
    c = z
    a += b
    if a != c
        return false
    end
    return true
end
```

A.1.2 Munin assembly code

```
set v0 i0
set v1 i1
set v2 i2
iadd v0 v1
set b0 0
cmp v0 v2
jon ne
set b0 1
end
```

A.2 LSPACE ADD algorithm

A.2.1 Pseudo-code

Note: it is assumed that the carry_flag is set automatically when binary addition is performed and is set to false at the start of each function.

```
function add(x, y, z)
    len = size_of(z)
    i = size_of(x)
    if i > len
        return false
    end
    i = size_of(y)
    if i > len_z
        return false
    end
```

```
        i = 0
        while i < len_z
            x_bit = get_nth_bit_of(x, i)
            y_bit = get_nth_bit_of(y, i)
            if carry_flag
                x_bit = add_bits(x_bit, 1)
            end
            x_bit = add_bits(x_bit, y_bit)
            z_bit = get_nth_bit_of(z, i)
            if x_bit != z_bit
                return false
            end
            i += 1
        end
        return true
    end
```

A.2.2 Munin assembly code

```
stl v00 i02
stl v01 i00
cmp v01 v00
jon g
jmp 7
set b00 0
end
stl v01 i01
cmp v01 v00
jon le
jmp 5
set v01 0
stnb v02 i00 v01
stnb v03 i01 v01
stnb v04 i02 v01
jon nc
badd v02 1
badd v02 v03
cmp v04 v02
jon ne
jmp 23
set b00 0
end
iadd v01 1
```

```
cmp v01 v00
jon ge
jmp 12
set b00 1
end
```

A.3 LSPACE PAL-ADD algorithm

A.3.1 Pseudo-code

Note: it is assumed that the carry_flag is set automatically when binary addition is performed and is set to false at the start of each function.

```
function pal_add(x, y)
    len = size_of(x)
    len_tmp = size_of(y)
    if len_tmp > len
        len = len_tmp
    end
    carry_on_last = carry_on_last_of_sum(x, y)
    i = 0
    while i < len
        i_bit = get_nth_bit_of_sum(x, y, i)
        j = len - i - 1 + carry_on_last
        j_bit = get_nth_bit_of_sum(x, y, j)
        if i_bit != j_bit
            return false
        end
        i += 1
    end
    return true
end

function get_nth_bit_of_sum(x, y, n)
    i = 0
    while i < n
        x_bit = get_nth_bit_of(x, i)
        y_bit = get_nth_bit_of(y, i)
        if carry_flag
            x_bit = add_bits(x_bit, 1)
        end
        x_bit = add_bits(x_bit, y_bit)
        i += 1
    end
    return x_bit
end
```

```
    function carry_on_last_of_sum(x, y)
        len = size_of(x)
        len_tmp = size_of(y)
        if len_tmp > len
            len = len_tmp
        end
        i = 0
        while i < len
            x_bit = get_nth_bit_of(x, i)
            y_bit = get_nth_bit_of(y, i)
            if carry_flag
                x_bit = add_bits(x_bit, 1)
            end
            x_bit = add_bits(x_bit, y_bit)
            i += 1
        end
        if carry_flag
            return 1
        else
            return 0
        end
    end
```

A.3.2 Munin assembly code

```
stl v00 i00
stl v01 i01
cmp v01 v00
jon l
set v00 v01
set v02 0
jmp 32
set v06 0
cmp v06 v00
jon l
jmp 30
set v07 0
set v08 v06
set v09 0
jmp 47
set v13 v09
set v08 v00
```

```
isub v08 v06
isub v08 1
iadd v08 v02
set v07 1
jmp 47
set v14 v09
cmp v13 v14
jon ne
jmp 28
set b00 0
end
iadd v06 1
jmp 8
set b00 1
end
set v03 0
stnb v04 i00 v03
stnb v05 i01 v03
jon nc
badd v04 1
badd v04 v05
iadd v03 1
cmp v03 v00
jon ge
jmp 33
set v02 0
jon nc
set v02 1
clf
jmp 7
clf
set v10 0
stnb v11 i00 v10
stnb v12 i01 v10
jon nc
badd v11 1
badd v11 v12
iadd v10 1
cmp v10 v08
jon g
jmp 49
set v09 v11
cmp v07 0
jon ne
```

```
jmp 15
jmp 22
```

B Munin assembly reference

Munin assembly reference				
Operator	Operand 1	Operand 2	Operand 3	Description
Assignment operaions				
set	D	S		Sets variable D equal to the value of S
stl	D	S		Sets variable D equal to the length of S
stnb	D	S	N	Sets variable D equal to the Nth bit of S
Integer arithmetic operaions				
iadd	D	S		Sets variable D equal to the value of D + S
isub	D	S		Sets variable D equal to the value of D - S
Binary arithmetic operaions				
badd	D	S		Sets one-bit variable D equal to the value of the binary sum of one-bit D and one-bit S; sets carry flag
bsub	D	S		Sets one-bit variable D equal to the value of the binary subtraction of one-bit D and one-bit S; sets the underflow flag
bsr	D	S		Sets variable D equal to the value of D « S
bsl	D	S		Sets variable D equal to the value of D » S
Comparison operaions				
cmp	A	B		Sets the equal flag if A == B ; sets the greater flag if A > B
clf				Clears all flags
Program flow operaions				
jmp	D			Jumps to line D
end				Ends the program