Sunny Patel

Prof. Johnson

Cs281 – Written Assignment 4

* 1. Octal(62) x Octal(12) = 110010 x 1010

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step | Action | Multiplier | Multiplicand | Product |
| 0 | Init vals | 001 010 | 000 000 110 010 | 0 |
| 1 | 1: 0 🡺 No operation | 001 010 | 000 000 110 010 | 0 |
| 1 | 2: Shift left Multiplicand | 001 010 | 000 001 100 100 | 0 |
| 1 | 3 :Shift right Multiplier | 000 101 | 000 001 100 100 | 0 |
| 2 | 1a: 1 🡺Prod = Prod + Mcand | 000 101 | 000 001 100 100 | 000 001 100 100 |
| 2 | 2: Shift left Multiplicand | 000 101 | 000 011 001 000 | 000 001 100 100 |
| 2 | 3: Shift right multiplier | 000 010 | 000 011 001 000 | 000 001 100 100 |
| 3 | 1: 0 🡺 No op | 000 010 | 000 011 001 000 | 000 001 100 100 |
| 3 | 2: Shift left Muliplicand | 000 010 | 000 110 010 000 | 000 001 100 100 |
| 3 | 3: Shift right Multiplier | 000 001 | 000 110 010 000 | 000 001 100 100 |
| 4 | 1: 1 🡺 Prod = Prod + Mcand | 000 001 | 000 110 010 000 | 000 111 110 100 |
| 4 | 2: Shift left Multiplicand | 000 001 | 001 100 100 000 | 000 111 110 100 |
| 4 | 3: Shift right Multiplier | 000 000 | 001 100 100 000 | 000 110 110 100 |
| 5 | 1: 0 🡺 No op | 000 000 | 001 100 100 000 | 000 110 110 100 |
| 5 | 2: Shift left Multiplicand | 000 000 | 001 100 100 000 | 000 110 110 100 |
| 5 | 3: Shift right Multiplier | 000 000 | 001 100 100 000 | 000 110 110 100 |
| 6 | 1: 0 🡺 No op | 000 000 | 001 100 100 000 | 000 110 110 100 |
| 6 | 2: Shift left Multiplicand | 000 000 | 001 100 100 000 | 000 110 110 100 |
| 6 | 3: Shift right Multiplier | 000 000 | 001 100 100 000 | 1. 0 110 100 |

1.2 MIPs psuedocode

T1 = 1 -- track loop iter.

Start loop main

If (get the last bit of B) = 1 then

Product = product + A

End if

Sll B B 1

SRL A A 1

If (T1 = 8) Then

Jump end

Else

Jump main

1.3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Step | Action | Multiplier | Multiplicand | Product | Sign |
| 0 | Init vals | 011 011 | 000 000 100 001 | 0 |  |
| 0 | Multiplier.sign XOR Multiplicand.sign (0 XOR 1) |  |  |  | 1 |
| 0 | Make positive | 011 011 | 000 000 000 001 | 0 |  |
| 1 | 1: 1 🡺 Prod = Prod + MCand | 011 011 | 000 000 000 001 | 000 000 000 001 |  |
| 1 | 2: Shift left Multiplicand | 011 011 | 000 000 000 010 | 000 000 000 001 |  |
| 1 | 3 :Shift right Multiplier | 001 101 | 000 000 000 010 | 000 000 000 001 |  |
| 2 | 1a: 1 🡺Prod = Prod + Mcand | 001 101 | 000 000 000 010 | 000 000 000 011 |  |
| 2 | 2: Shift left Multiplicand | 001 101 | 000 000 000 100 | 000 000 000 011 |  |
| 2 | 3: Shift right multiplier | 000 110 | 000 000 000 100 | 000 000 000 011 |  |
| 3 | 1: 0 🡺 No op | 000 110 | 000 000 000 100 | 000 000 000 011 |  |
| 3 | 2: Shift left Muliplicand | 000 110 | 000 000 001 000 | 000 000 000 011 |  |
| 3 | 3: Shift right Multiplier | 000 011 | 000 000 001 000 | 000 000 000 011 |  |
| 4 | 1: 1 🡺 Prod = Prod + Mcand | 000 011 | 000 000 001 000 | 000 000 001 011 |  |
| 4 | 2: Shift left Multiplicand | 000 011 | 000 000 010 000 | 000 000 001 011 |  |
| 4 | 3: Shift right Multiplier | 000 001 | 000 000 010 000 | 000 000 001 011 |  |
| 5 | 1: 1 🡺 Prod = Prod + Mcand | 000 001 | 000 000 010 000 | 000 000 011 011 |  |
| 5 | 2: Shift left Multiplicand | 000 001 | 000 000 100 000 | 000 000 011 011 |  |
| 5 | 3: Shift right Multiplier | 000 000 | 000 000 100 000 | 000 000 011 011 |  |
| 6 | 1: 0 🡺 No op | 000 000 | 000 000 100 000 | 000 000 011 011 |  |
| 6 | 2: Shift left Multiplicand | 000 000 | 000 000 100 000 | 000 000 011 011 |  |
| 6 | 3: Shift right Multiplier | 000 000 | 000 000 100 000 | 000 000 011 011 |  |
| 7 | Prod msb = sign | 000 000 | 000 000 100 000 | 000 000 111 011 |  |

2.1 Assuming A = 33(base 16) and B = 55(base 16)

33 = 51(base 10)

= 32 + 18 + 1

So then 33 x 55 can be calculated by shifting 55 by 5 + shifting 55 by 4 + shifting 55 by 1 + shifting 55 by 0. A total of 4 shifts and 3adds.

2.2) Assuming A=33(base 16) B=55(base 16)

A = 51 (base 10) and B = 85(base 10)

110 011 1010101

Sign bit to positive gives: 010 011 and 0010101 respectively. Sign reg is 0 since (1 xor 1)

Then 19 (110 011) = 16 + 2 + 1, so shift 21 by 4 + shift 21 by 1 + shift 21 by 0.

2.3) MIPS implementation

Result = 0

Loop

Shift multiplier left until the rightmost is lined up with the leftmost 1 in the multiplicand.

Add the multiplier in that exact position to the product

Remove the right most one in the multiplicand.

If the multiplicand = 0 then END

Else LOOP

2.4) Booth’s Algorithm

Booth's algorithm will require two extra 1-bit registers that is Current Bit, and Previous Bit.

First you want to populate the product register with a zero value in the upper half and the multiplier in the lower half. Populate the multiplicand. Populate the Previous bit with a 0 since this is our first step.

Then Populate the previous bit and check the least significant bit of the product register. Perform a operation according to what the Current and Previous bit hold. If the both bits are either 1 or 0 do nothing, if current is 0 and previous is 1 then Add multiplicand to upper Product register. If current is 1 and previous is 0 then Subtract multiplicand from the upper Product register.

After that do a right shift of the product register, and then repeat the last two steps. Repeat until you have looped at least the same number of times as the number of bits in multiplier.

2.5) Assuming the A=72(octal) and B=07(octal)

A= 58(deci) and B = 7 (deci)

A=(111 010) and B=(000 111) ; -A = 000 110

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step | Product Register | Multiplicand | Current Bit | Previous Bit |
| 0 | 000000 000111 | 111 010 | 1 | 0 |
| 1 | 000110 000011 | 111 010 | 1 | 1 |
| 2 | 000110 000001 | 111 010 | 1 | 1 |
| 3 | 000110 000000 | 111 010 | 0 | 1 |
| 4 | 1000000 000000 | 111 010 | 0 | 0 |
| 5 | 100000 000000 |  |  |  |

3.1)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step | Operation | Quotient | Divisor | Remainder |
| 0 | Init. | 000000 | 010001 000000 | 000000 111100 |
| 1 | 1. Rem = Rem - Div |  |  | 101111 111100 |
| 1 | 2B: Rem<0  Restore by adding  Div to rem shift Q right |  | 001000 100000 | 000000 111100 |
| 1 | 3: Shift right div |  | 001000 100000 |  |
| 2 | 1. Rem = Rem - Div |  | 0010001 100000 | 111000 011100 |
| 2 | 1. Rem < 0 |  | 000100 010000 | 000000 111100 |
| 2 | 1. Shift right div |  | 000100 010000 |  |
| 3 | 1. Rem = Rem -Div |  | 000100 010000 | 111100 101100 |
| 3 | 1. Rem < 0 |  |  | 000000 111100 |
| 3 | 1. Shift right div |  | 000010 001000 |  |
| 4 | 1. Rem = Rem - Div |  | 000010 001000 | 111110 110100 |
| 4 | 1. Rem < 0 |  |  | 000000 111100 |
| 4 | 1. Shift right div |  | 000001 000100 | 000000 111100 |
| 5 | 1. Rem = Rem - Div |  |  | 111110 111100 |
| 5 | 1. Rem < 0 |  |  | 000000 111100 |
| 5 | 1. Shift right div |  | 000000 000010 | 000000 111100 |
| 6 | 1. Rem = Rem - Div |  | 000000 000010 | 111111 111110 |
| 6 | 1. Rem < 0 |  |  | 000000 111100 |
| 6 | 1. Shift right div |  | 000000 000001 |  |