! CPSC 355 ASSIGNMENT 2A: Bitwise Logical and Shift Operations

! ============================================================

! Completed on 10-10-2013 by HUNG, Michael (UCID: 10099049)

!

! This program simulates a CRC for the input 0xaaaa8C01 0xFF001234 0x13579BDF 0xC8B4AE32

! and generates a checksum.

output: .asciz "Input data: 0xaaaa8C01 0xFF001234 0x13579BDF 0xC8B4AE32\nChecksum (before): %x Checksum (after): %x\n" ! Output string for printing data used to feed checksum, and checksums themselves

.align 4

.global main

main: save %sp, -96, %sp

clr %l0 ! Initialize register for checksum

clr %l4 ! Initialize register for data input 1

clr %l5 ! Initialize register for data input 2

clr %l6 ! Initialize register for data input 3

clr %l7 ! Initialize register for data input 4

set 0xFFFF, %l0 ! Prepare checksum

set 0xaaaa8C01, %l4 ! Prepare data input 1

set 0xFF001234, %l5 ! Prepare data input 2

set 0x13579BDF, %l6 ! Prepare data input 3

set 0xC8B4AE32, %l7 ! Prepare data input 4

mov 0, %i0 ! Initialize counter for loops

dataA: inc %i0 ! loop to process data input 1

set 0x8000, %i5 ! Avoiding relocation errors

and %l0, %i5, %i1 ! %i1 contains the bit to be checked

! against b0, b5, and b12

srl %i1, 15, %i1

and %l0, 0x800, %i2 ! %i2 contains one of the bits to be

! checked against %i1

srl %i2, 11, %i2

and %l0, 0x10, %i3 ! %i3 contains one of the bits to be

! checked against %i1

srl %i3, 4, %i3

set 0x80000000, %i5 ! Avoiding relocation errors

and %l4, %i5, %i4 ! %i4 contains the bit of the data to be

! fed into the CRC to be checked

srl %i4, 31, %i4

sll %l4, 1, %l4 ! Shift the data in %l4 to prepare for

! next iteration

sll %l0, 1, %l0 ! %l0 contains the newly shifted checksum

xor %i1, %i2, %i2 ! xor b15 and b11

xor %i1, %i3, %i3 ! xor b15 and b4

xor %i1, %i4, %i4 ! xor b15 and b31

srl %l0, 13, %i5 ! Splitting checksum in order to update

! b12; Ran out of registers, so no more bitmasks

sll %i5, 1, %i5

add %i5, %i2, %i5 ! Updating the first split with xor result

sll %i5, 12, %i5 ! Shifting first split back to the left

! for addition with other split

and %l0, 0xFFF, %o5 ! Preparing second part of split for recombination

add %i5, %o5, %l0 ! Putting the splits back together with updated b12

clr %i5 ! Resetting registers for more splitting

clr %o5

srl %l0, 6, %i5 ! Splitting checksum in order to update b5

sll %i5, 1, %i5

add %i5, %i3, %i5

sll %i5, 5, %i5

and %l0, 0x1F, %o5 ! Preparing second part of split for recombination

add %i5, %o5, %l0 ! Putting the splits back together with updated b5

clr %i5 ! Resetting placeholder registers

clr %o5

add %l0, %i4, %l0 ! Update checksum for b0

cmp %i0, 32 ! Has the loop iterated 32 times for 32 bits?

ble dataA

nop

clr %i0 ! Reset counter and bit-holding registers

clr %i1

clr %i2

clr %i3

clr %i4

ba dataB

nop

dataB: inc %i0 ! loop to process data input 2

set 0x8000, %i5 ! Avoiding relocation errors

and %l0, %i5, %i1 ! %i1 contains the bit to be checked

! against b0, b5, and b12

srl %i1, 15, %i1

and %l0, 0x800, %i2 ! %i2 contains one of the bits to be

! checked against %i1

srl %i2, 11, %i2

and %l0, 0x10, %i3 ! %i3 contains one of the bits to be

! checked against %i1

srl %i3, 4, %i3

set 0x80000000, %i5 ! Avoiding relocation errors

and %l5, %i5, %i4 ! %i4 contains the bit of the data to be

! fed into the CRC to be checked

srl %i4, 31, %i4

sll %l5, 1, %l5 ! Shift the data in %l4 to prepare

! for next iteration

sll %l0, 1, %l0 ! %l0 contains the newly shifted checksum

xor %i1, %i2, %i2 !

xor %i1, %i3, %i3 !

xor %i1, %i4, %i4 !

srl %l0, 13, %i5 ! Splitting checksum in order to update b12

sll %i5, 1, %i5

add %i5, %i2, %i5

sll %i5, 12, %i5

and %l0, 0xFFF, %o5 ! Preparing second part of split for recombination

add %i5, %o5, %l0 ! Putting the splits back together with updated b12

clr %i5 ! Resetting registers for more splitting

clr %o5

srl %l0, 6, %i5 ! Splitting checksum in order to update b5

sll %i5, 1, %i5

add %i5, %i3, %i5

sll %i5, 5, %i5

and %l0, 0x1F, %o5 ! Preparing second part of split for recombination

add %i5, %o5, %l0 ! Putting the splits back together with updated b5

clr %i5 ! Resetting placeholder registers

clr %o5

add %l0, %i4, %l0 ! Update checksum for b0

cmp %i0, 32 ! Has the loop iterated 32 times for 32 bits?

ble dataB

nop

clr %i0 ! Reset counter and bit-holding registers

clr %i1

clr %i2

clr %i3

clr %i4

ba dataC

nop

dataC: inc %i0 ! loop to process data input 3

set 0x8000, %i5 ! Avoiding relocation errors

and %l0, %i5, %i1 ! %i1 contains the bit to be

! checked against b0, b5, and b12

srl %i1, 15, %i1

and %l0, 0x800, %i2 ! %i2 contains one of the bits to be

! checked against %i1

srl %i2, 11, %i2

and %l0, 0x10, %i3 ! %i3 contains one of the bits to be

! checked against %i1

srl %i3, 4, %i3

set 0x80000000, %i5 ! Avoiding relocation errors

and %l6, %i5, %i4 ! %i4 contains the bit of the data to

! be fed into the CRC to be checked

srl %i4, 31, %i4

sll %l6, 1, %l6 ! Shift the data in %l4 to prepare for

! next iteration

sll %l0, 1, %l0 ! %l0 contains the newly shifted checksum

xor %i1, %i2, %i2 !

xor %i1, %i3, %i3 !

xor %i1, %i4, %i4 !

srl %l0, 13, %i5 ! Splitting checksum in order to update b12

sll %i5, 1, %i5

add %i5, %i2, %i5

sll %i5, 12, %i5

and %l0, 0xFFF, %o5 ! Preparing second part of split for recombination

add %i5, %o5, %l0 ! Putting the splits back together with updated b12

clr %i5 ! Resetting registers for more splitting

clr %o5

srl %l0, 6, %i5 ! Splitting checksum in order to update b5

sll %i5, 1, %i5

add %i5, %i3, %i5

sll %i5, 5, %i5

and %l0, 0x1F, %o5 ! Preparing second part of split for recombination

add %i5, %o5, %l0 ! Putting the splits back together with updated b5

clr %i5 ! Resetting placeholder registers

clr %o5

add %l0, %i4, %l0 ! Update checksum for b0

cmp %i0, 32 ! Has the loop iterated 32 times for 32 bits?

ble dataC

nop

clr %i0 ! Reset counter and bit-holding registers

clr %i1

clr %i2

clr %i3

clr %i4

ba dataD

nop

dataD: inc %i0 ! loop to process data input 4

set 0x8000, %i5 ! Avoiding relocation errors

and %l0, %i5, %i1 ! %i1 contains the bit to be checked

! against b0, b5, and b12

srl %i1, 15, %i1

and %l0, 0x800, %i2 ! %i2 contains one of the bits to be

! checked against %i1

srl %i2, 11, %i2

and %l0, 0x10, %i3 ! %i3 contains one of the bits to be

! checked against %i1

srl %i3, 4, %i3

set 0x80000000, %i5 ! Avoiding relocation errors

and %l7, %i5, %i4 ! %i4 contains the bit of the data

! to be fed into the CRC to be checked

srl %i4, 31, %i4

sll %l7, 1, %l7 ! Shift the data in %l4 to prepare

! for next iteration

sll %l0, 1, %l0 ! %l0 contains the newly shifted checksum

xor %i1, %i2, %i2 !

xor %i1, %i3, %i3 !

xor %i1, %i4, %i4 !

srl %l0, 13, %i5 ! Splitting checksum in order to update b12

sll %i5, 1, %i5

add %i5, %i2, %i5

sll %i5, 12, %i5

and %l0, 0xFFF, %o5 ! Preparing second part of split for recombination

add %i5, %o5, %l0 ! Putting the splits back together with updated b12

clr %i5 ! Resetting registers for more splitting

clr %o5

srl %l0, 6, %i5 ! Splitting checksum in order to update b5

sll %i5, 1, %i5

add %i5, %i3, %i5

sll %i5, 5, %i5

and %l0, 0x1F, %o5 ! Preparing second part of split for recombination

add %i5, %o5, %l0 ! Putting the splits back together with updated b5

clr %i5 ! Resetting placeholder registers

clr %o5

add %l0, %i4, %l0 ! Update checksum for b0

cmp %i0, 32 ! Has the loop iterated 32 times for 32 bits?

ble dataD

nop

clr %i0 ! Reset counter and bit-holding registers

clr %i1

clr %i2

clr %i3

clr %i4

ba print

nop

print: set output, %o0 ! Prepare printf output string

set 0xFFFF, %o1 ! Checksum before feeding data will always be 0xFFFF

and %o1, %l0, %o2 ! Get only the last 16 bits of the

! checksum after feeding data

call printf ! Print checksum message

nop

mov 1, %g1

ta 0