### EEEN 3449

### Microprocessor Systems

# Bit Testing

Tyler Hurson

Spring 2017

**I. INTRODUCTION**

* 1. **Purpose**

The purpose of this experiment is explore the various ways of checking and counting individuals bits using the Assembly language.

* 1. **Problem**

Rotate and branching instructions were used to achieve the desired results in program A (Appendix A). A 2-byte number was rotated to the right a total of 16 times (iterating over each bit). Each iteration, the value of the carry flag (C) was checked to determine if there was a 1 that was rotated off. If C = 1, a counter was incremented. At the end of the program, the value of the counter was the number of 1s in the number.

Program B determined if each number in an array was divisible by 4. If it was, the number was moved to a new location. Else, it was moved to a different location. A mask branch instruction was used to determine the divisibility of numbers. For each iteration, the number was compared to $03 (binary: 0000 0011) to determine if the least two significant bits were set.

Program C determined if each number in array was divisible by 3. If it was, the number was moved to a new location. Else, it was moved to a different location. Instead of mask branch instruction, a division instruction was used. Each iteration, the number was divided by 3, and then the remainder was checked.

* 1. **Scope**

The scope of this experiment is limited to the HCS12 microcontroller. Several instructions will be used from the HCS12 instruction set.

**II. TEST AND EVALUATION**

**2.1 Apparatus**

The equipment used in this test includes: Dragon12-Junior development board, USB power cord, and laptop PC with AsmIDE.

**2.2 Procedure**

1. The development board was connected to the computer.
2. The COM port number was determined under Device Manager on PC. AsmIDE was launched. Under View -> Options -> COM Port, the COM port was set to the device’s number. The Terminal Window was enabled. Under Set COM Options, the default values were restored.
3. Program A was opened, and then assembled. After no errors were recorded, program A was downloaded into the development board, by typing load in the Terminal Window in AsmIDE, then downloading the program.
4. g 2000 was typed to execute the program. At the end of the program, md 1505 was typed to confirm that the counter (stored at $1505) was correct.
5. Program B was opened, and then assembled. After no errors were recorded, program B was downloaded into the development board.
6. g 2000 was typed to execute the program. At the end of the program, md 1500 was typed to confirm that it contained all the numbers divisible by 4. md 1520 was typed to confirm that it contained all the numbers not divisible by 4.
7. Program C was opened, and then assembled. After no errors were recorded, program C was downloaded into the development board.
8. g 2000 was typed to execute the program. At the end of the program, md 1500 was typed to confirm that it contained all the numbers divisible by 3. md 1520 was typed to confirm that it contained all the numbers not divisible by 3.

**III. RESULTS**

* 1. **Data**

Table 1 displays the final result of program A after execution. $1505 contains the 1’s counter.

Table 1: Final Result of Program A

|  |  |
| --- | --- |
| **ADDRESS** | **CONTENT** |
| $1500 | 12 |
| $1501 | 34 |
| $1502 | CF |
| $1503 | 23 |
| $1504 | 79 |
| $1505 | 05 |

Table 2 displays the final result of program B after execution. $1500 contains the numbers divisible by 4, $1540 contains the numbers not divisible by 4.

Table 2: Final Result of Program B

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ADDRESS** | **CONTENT** | **ADDRESS** | **CONTENT** | **ADDRESS** | **CONTENT** |
| $1500 | 1C | $1510 | DE | $1520 | 01 |
| $1501 | 4C | $1511 | 45 | $1521 | 03 |
| $1502 | 14 | $1512 | AA | $1522 | 05 |
| $1503 | 40 | $1513 | 58 | $1523 | 06 |
| $1504 | 79 | $1514 | 5B | $1524 | 13 |
| $1505 | 05 | $1515 | E6 | $1525 | 29 |
| $1506 | 59 | $1516 | C3 | $1526 | 35 |
| $1507 | 25 | $1517 | 01 | $1527 | 0D |
| $1508 | BB | $1518 | 42 | $1528 | 2A |
| $1509 | 93 | $1519 | 10 | $1529 | 0E |
| $150A | 94 | $151A | 8D | $152A | 36 |
| $150B | 00 | $151B | 8B | $152B | 4A |
| $150C | 89 | $151C | 94 | $152C | 1D |
| $150D | 43 | $151D | BC | $152D | 21 |
| $150E | CB | $151E | 18 | $152E | 29 |
| $150F | D3 | $151F | 90 | $152F | 2D |

Table 3 displays the final result of program c after execution. $1500 contains the numbers divisible by 3, $1540 contains the numbers not divisible by 3.

Table 3: Final Result of Program C

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ADDRESS** | **CONTENT** | **ADDRESS** | **CONTENT** | **ADDRESS** | **CONTENT** |
| $1500 | 1C | $1510 | DE | $1520 | 01 |
| $1501 | 4C | $1511 | 45 | $1521 | 03 |
| $1502 | 14 | $1512 | AA | $1522 | 05 |
| $1503 | 40 | $1513 | 58 | $1523 | 06 |
| $1504 | 79 | $1514 | 5B | $1524 | 13 |
| $1505 | 05 | $1515 | E6 | $1525 | 29 |
| $1506 | 59 | $1516 | C3 | $1526 | 35 |
| $1507 | 25 | $1517 | 01 | $1527 | 0D |
| $1508 | BB | $1518 | 42 | $1528 | 2A |
| $1509 | 93 | $1519 | 10 | $1529 | 0E |
| $150A | 94 | $151A | 8D | $152A | 36 |
| $150B | 00 | $151B | 8B | $152B | 4A |
| $150C | 89 | $151C | 94 | $152C | 1D |
| $150D | 43 | $151D | BC | $152D | 21 |
| $150E | CB | $151E | 18 | $152E | 29 |
| $150F | D3 | $151F | 90 | $152F | 2D |

* 1. **Analysis**

Figure 4: Flowchart of Program A

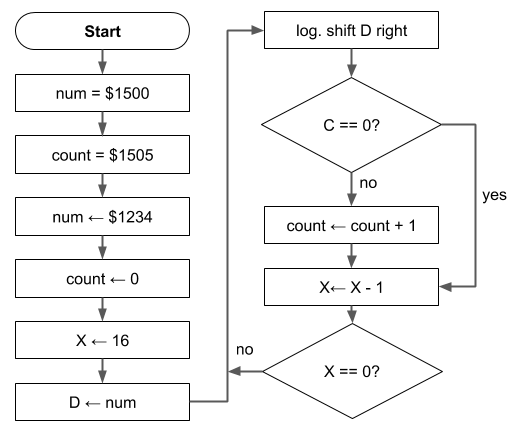


Figure 5: Flowchart of Program B

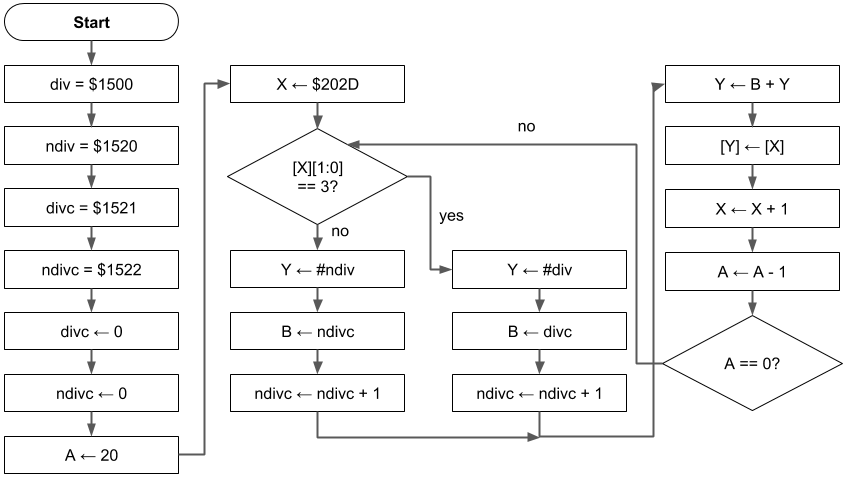
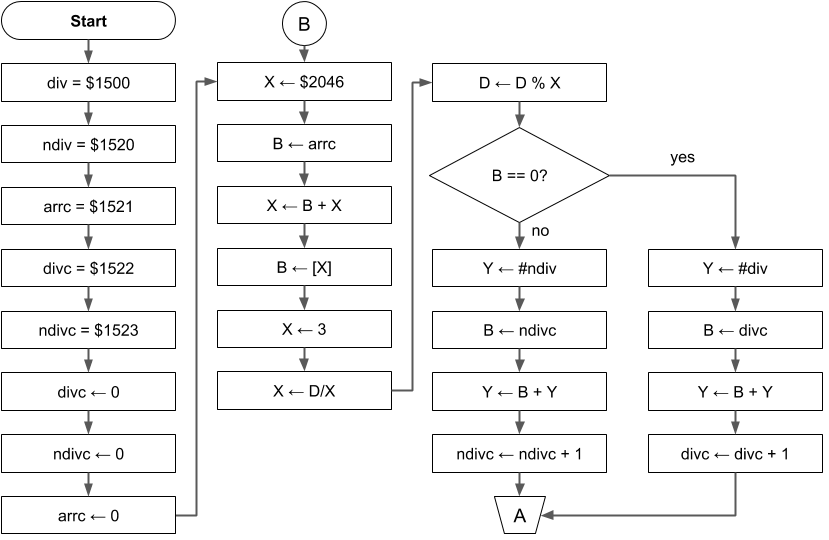
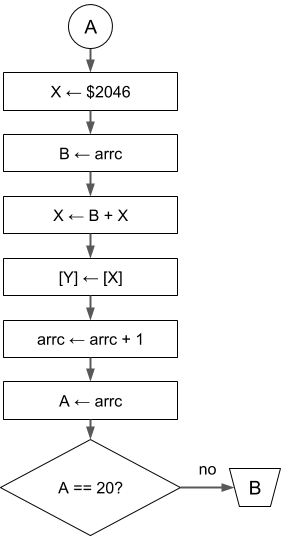


Figure 6: Flowchart of Program C

****

****

1. **CONCLUSION**

**4.1 Assessment**

This experiment served as an introduction to bit checking/counting. In part A, the rotate instruction was used to determine the number of 1s in a generic 2-byte number. In part B, a mask was used to determine if a generic number was divisible by 4. In part C, the division instruction was used to determine if a generic number was divisible by 3.

**APPENDIX A**

**ASSEMBLY PROGRAM A**

N equ 16

org $1500

num rmb 2

org $1505

count rmb 1 ; 1s count

org $2000 ; program start

movw #$1234,$1500

clr count ; count = 0

ldx #N ; initialize loop counter to 16

ldd num ; D = $1500

loop lsrd ; logical right shift D

bcc zero ; branch to zero if carry flag = 0

inc count ; increment 1s counter

zero dbne X, loop ; decrement X, branch if X == 0

swi

end

**APPENDIX B**

**ASSEMBLY PROGRAM B**

N equ 20

org $1500

div rmb $20 ; 32 bytes for divisible-by-4 array

ndiv rmb $20 ; 32 bytes for not-divisible-by-4 array

divc rmb 1 ; 1 byte for divisible counter

ndivc rmb 1 ; 1 byte for non-divisible counter

org $2000

movw #0000, divc ; clear both array counters

ldaa #N ; A = array size

ldx #array ; load address of main array into X

loop brclr 0,X,$03,isdiv ; branch if equal to mask

ldy #ndiv ; Y = non-divisible array address

ldab ndivc ; B = non-divisible array counter

inc ndivc ; increment non-divisible counter

bra done ; skip over true block

isdiv ldy #div ; load address of divisible array to Y

ldab divc ; load counter of divisible array to B

inc divc ; increment divisible counter

done aby ; Y = Y + B

movb 1,X+,0,Y ; move number, increment X

dbne A, loop ; decrement A, branch to loop if A != 0

swi

array db 1, 3, 5, 6, 19, 41, 53, 28, 13, 42, 76, 14, 20, 54, 64, 74, 29, 33, 41, 45

end

**APPENDIX C**

**ASSEMBLY PROGRAM C**

N equ 20 ; array size

qnt equ 3 ; divisor

org $1500

div rmb $20 ; reserve 32 bytes for divisible-by-3 array

ndiv rmb $20 ; reserve 32 bytes for not-divisible-by-3 array

arrc rmb 1 ; reserve 1 byte for main array counter

divc rmb 1 ; reserve 1 byte for divisible counter

ndivc rmb 1 ; reserve 1 byte for non-divisible counter

org $2000

fill #00, 3 ; clear array counters

loop ldx #array ; load pointer to main array into X

ldab arrc ; B = array counter

abx ; B = B + X

clra ; A = 0

ldab 0,X ; B = m[X]

ldx #qnt ; X = 3

idiv ; X = D/X, D = D % X

cmpb #0 ; compare least significant byte in D to 0

beq isdiv ; if remainder == 0, goto isdiv branch

ldy #ndiv ; load pointer to non-divisible array into Y

ldab ndivc ; load non-divisible counter into B

aby ; Y = B + Y

inc ndivc ; increment non-divisible counter

bra done ; branch to done, skip over true block

isdiv ldy #div ; load pointer to divisible array into Y

ldab divc ; load divisible counter into B

aby ; Y = B + Y

inc divc ; increment divisible counter

done ldx #array ; load pointer to main array into X again

ldab arrc ; load main array counter into B

abx ; X = B + X

movb 0,X,0,Y ; move current element to designated array

inc arrc ; increment main array pointer

ldaa arrc ; A = main array pointer

cmpa #N ; compare A to array size

bne loop ; loop if A < size

swi

array db 1, 3, 5, 6, 19, 41, 53, 28, 13, 42, 76, 14, 20, 54, 64, 74, 29, 33, 41, 45

end