EEEN 3449

Microprocessor Systems

Condition Flags and Rotate Instruction

Tyler Hurson

Spring 2017

Texas A&M University – Kingsville, Electrical Engineering and Computer Science Department

I. INTRODUCTION

1.1 Purpose

The purpose of this experiment is to explore how the rotate, shift, boolean, and arithmetic instructions in the Assembly language affect the conditional code register (CCR).

1.2 Problem

Rotate, shift, boolean, and arithmetic instructions were used in order to manipulate the conditional code register (CCR). The CCR is a 1-byte register consisting of eight 1-bit flags in the format SXHINZVC, where C is the least significant bit. For this experiment, only the least 4 significant bits will be considered. C = carry flag, V = overflow flag, Z = zero flag, N = negative flag.

The carry flag is set when a carry occurs during addition or a borrow occurs during subtraction. Shift and rotate instructions also operate through the carry flag. The overflow flag is set when a two's complement overflow occurs as a result of an operation. The zero flag is set when the result of an operation is zero. The negative flag is set when the most significant bit of the result is 1.

In Program A (Appendix A), the CCR was initially set to xxxx0101 (where x = don't care). The zero and carry flag were set using the `orcc` instruction. The overflow and negative flag were cleared using the `andcc` instruction. $50 was loaded into A using the `ldaa` instruction, and then 40 was subtracted from A using the `suba` instruction. The value of A is tested using the `tsta` instruction. Then, $50 was added to A using the `adda` instruction. Finally, A was shifted right once using the `lsra` instruction, then rotated left using the `rola` instruction, and then shifted left using the `lsla` instruction. Throughout the execution of the program, the value of the CCR changed.

In Program B (Appendix B), the CCR was initially set to xxxx0100. The zero flag was set. The overflow, carry, and negative flag were. $50 was loaded into, and then 40 was subtracted from A. The value of A is. Then, $40 was added to. $78 was added to A again. A was shifted right, then rotated left, and then shifted left. Finally, $CF was added to A. Again, the value of the CCR changed throughout the execution of the program.

In Program C (Appendix C), $8287 was loaded into D using the `ldd` instruction. Then, $8998 was added to D using the `addd` instruction. Next, the value of D was stored at $1502 using the `std` instruction. Then, $95 was loaded into A. A was added to the value of the carry flag, then added to $45, and then stored in A using the `adca` instruction. This value was stored in $1501. Finally, $A2 was loaded into A again. A was added to the value of the carry flag, then added to $78, and stored back into A. A was then stored in $1500. The value of the CCR changed throughout the execution of the program.

In Program D (Appendix D), $4321 was loaded into D. $6789 was subtracted from D using the `subd` instruction. Then, the value of D was stored in $1502. $65 was loaded to A. The value of the carry flag and $45 were subtracted from A using the `sbca` instruction. A was stored in $1501. Finally, $87 was stored in A. The value of the carry flag and $23 were subtracted from A. A was then stored in $1500. The value of the CCR changed throughout the execution of the program.

1.3 Scope

The scope of this experiment is limited to the HCS12 microcontroller. Only a few basic 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. `br 2005` was typed to set a breakpoint. `g 2000` was typed to execute the program. `t 10` was typed repeatedly to trace the program, line by line, until the program ended. As the program was traced, the values in the CPU registers were recorded.
5. Program B was opened, and then assembled. After no errors were recorded, program B was downloaded into the development board, by typing `load` in the Terminal Window in AsmIDE, then downloading the program.

6. `br 2005` was typed to set a breakpoint. `g 2000` was typed to execute the program. `t 10` was typed repeatedly to trace the program, line by line, until the program ended. As the program was traced, the values in the CPU registers were recorded.
7. Program C was opened, and then assembled. After no errors were recorded, program C was downloaded into the development board, by typing `load` in the Terminal Window in AsmIDE, then downloading the program.
8. `br 2005` was typed to set a breakpoint. `g 2000` was typed to execute the program. `t 10` was typed repeatedly to trace the program, line by line, until the program ended. As the program was traced, the values in the CPU registers were recorded.
9. Program D was opened, and then assembled. After no errors were recorded, program D was downloaded into the development board, by typing `load` in the Terminal Window in AsmIDE, then downloading the program.
10. `br 2005` was typed to set a breakpoint. `g 2000` was typed to execute the program. `t 10` was typed repeatedly to trace the program, line by line, until the program ended. As the program was traced, the values in the CPU registers were recorded.

III. RESULTS

3.1 Data

For each program, the value of each CPU register, including the CCR, was recorded. The full results for each program appear in their respective Appendix below.

III. CONCLUSION

4.1 Assessment

This experiment served as an introduction to the rotate and shift instructions in the Assembly language. Also explored were how these instructions, as well as arithmetic instructions, affected the value of the conditional code register (CCR). It was discovered that the flags in the CCR were set and cleared according to the conditions specified.

APPENDIX A

ASSEMBLY PROGRAM A

```
org     $1500   ; program start
orcc    #$05    ; Set zero and carry flag
andcc   #$F5    ; Clear overflow and negative flag
ldaa    #$50    ; A = $50
suba    #40     ; A = A - 40
tsta
adda    #$50    ; A = A + $50
lsra            ; logical shift right A
rola            ; rotate A left
lsla            ; logical shift left A
swi
end
```

ORCC #$05

PC	SP	X	Y	A	B	SXHI NZVC
1502	3C00	0000	0000	00	00	1001 0000

ANDCC #$F5

PC	SP	X	Y	A	B	SXHI NZVC
1504	3C00	0000	0000	00	00	1001 0101

LDAA #$50

PC	SP	X	Y	A	B	SXHI NZVC
1506	3C00	0000	0000	00	00	1001 0101

SUBA #$28

PC	SP	X	Y	A	B	SXHI NZVC
1508	3C00	0000	0000	50	00	1001 0001

TSTA

PC	SP	X	Y	A	B	SXHI NZVC
150A	3C00	0000	0000	28	00	1001 0000

ADDA #$50

PC	SP	X	Y	A	B	SXHI NZVC
150B	3C00	0000	0000	28	00	1001 0000

LSRA

PC	SP	X	Y	A	B	SXHI NZVC
150D	3C00	0000	0000	78	00	1001 0000

ROLA

PC	SP	X	Y	A	B	SXHI NZVC
150E	3C00	0000	0000	3C	00	1001 0000

ASLA

PC	SP	X	Y	A	B	SXHI NZVC
150F	3C00	0000	0000	78	00	1001 0000

SWI

PC	SP	X	Y	A	B	SXHI NZVC
1510	3C00	0000	0000	F0	00	1001 1010

APPENDIX B

ASSEMBLY PROGRAM B

```
org     $1500   ; program start
orcc    #$04    ; Set zero flag
andcc   #$F4    ; Clear overflow, negative, and carry flag
ldaa    #$50    ; A = $50
suba    #40     ; A = A - 40
tsta
adda    #$40    ; A = A + $40
adda    #$78    ; A = A + $78
lsra            ; logical shift right A
rola            ; rotate A left
adda    #$CF    ; logical shift left A
swi
end
```

ORCC #$04

PC	SP	X	Y	A	B	SXHI NZVC
1502	3C00	0000	0000	00	00	1001 0000

ANDCC #$F4

PC	SP	X	Y	A	B	SXHI NZVC
1504	3C00	0000	0000	00	00	1001 0100

LDAA #$50

PC	SP	X	Y	A	B	SXHI NZVC
1506	3C00	0000	0000	00	00	1001 0100

SUBA #$28

PC	SP	X	Y	A	B	SXHI NZVC
1508	3C00	0000	0000	50	00	1001 0000

TSTA

PC	SP	X	Y	A	B	SXHI NZVC
150A	3C00	0000	0000	28	00	1001 0000

ADDA #$40

PC	SP	X	Y	A	B	SXHI NZVC
150B	3C00	0000	0000	28	00	1001 0000

ADDA #$78

PC	SP	X	Y	A	B	SXHI NZVC
150D	3C00	0000	0000	68	00	1001 0000

LSRA

PC	SP	X	Y	A	B	SXHI NZVC
150F	3C00	0000	0000	E0	00	1011 1010

ROLA

PC	SP	X	Y	A	B	SXHI NZVC
1510	3C00	0000	0000	70	00	1011 0000

ADDA #$CF

PC	SP	X	Y	A	B	SXHI NZVC
1511	3C00	0000	0000	E0	00	1011 1010

SWI

PC	SP	X	Y	A	B	SXHI NZVC
1513	3C00	0000	0000	AF	00	1001 1001

APPENDIX C

ASSEMBLY PROGRAM C

```
        org     $1500   ; program start
        ldd     #$8287  ; D = $8287
        addd    #$8998  ; D += 8998
        std     $1502   ; $1502 = D
        ldaa    #$95    ; A = $95
        adca    #$45    ; A += carry + $45
        staa    $1501   ; $1501 = A
        ldaa    #$A2    ; A = $A2
        adca    #$78    ; A += carry + $78
        staa    $1500   ; $1500 = A
        swi
        end
```

LDD #$8287

PC	SP	X	Y	A	B	SXHI NZVC
1502	3C00	0000	0000	00	00	1001 0000

ADDD #$8998

PC	SP	X	Y	A	B	SXHI NZVC
1505	3C00	0000	0000	82	87	1001 1000

STD $1502

PC	SP	X	Y	A	B	SXHI NZVC
1508	3C00	0000	0000	0C	1F	1001 0011

LDAA #$95

PC	SP	X	Y	A	B	SXHI NZVC
150B	3C00	0000	0000	0C	1F	1001 0001

ADCA #$45

PC	SP	X	Y	A	B	SXHI NZVC
150D	3C00	0000	0000	95	1F	1001 1001

STAA $1501

PC	SP	X	Y	A	B	SXHI NZVC
150F	3C00	0000	0000	DB	1F	1001 1000

LDAA #$A2

PC	SP	X	Y	A	B	SXHI NZVC
1512	3C00	0000	0000	DB	1F	1001 1000

ADCA #$78

PC	SP	X	Y	A	B	SXHI NZVC
1514	3C00	0000	0000	A2	1F	1001 1000

STAA $1500

PC	SP	X	Y	A	B	SXHI NZVC
1516	3C00	0000	0000	1A	1F	1001 0001

SWI

PC	SP	X	Y	A	B	SXHI NZVC
1519	3C00	0000	0000	1A	1F	1001 0001

APPENDIX D

ASSEMBLY PROGRAM D

```
        org     $1500   ; program start
        ldd     #$4321  ; D = $4321
        subd    #$6789  ; D = D - $6789
        std     $1502   ; $1502 = D
        ldaa    #$65    ; A = $65
        sbca    #$45    ; A = A - carry - $45
        staa    $1501   ; $1501 = A
        ldaa    #$87    ; A = $87
        sbca    #$23    ; A = A - carry - $23
        staa    $1500   ; $1500 = A
        swi
        end
```

LDD #$4321

PC	SP	X	Y	A	B	SXHI NZVC
1502	3C00	0000	0000	00	00	1001 0000

SUBD #$6789

PC	SP	X	Y	A	B	SXHI NZVC
1505	3C00	0000	0000	43	21	1001 0000

STD $1502

PC	SP	X	Y	A	B	SXHI NZVC
1508	3C00	0000	0000	DB	98	1001 1001

LDAA #$65

PC	SP	X	Y	A	B	SXHI NZVC
150B	3C00	0000	0000	DB	98	1001 1001

SBCA #$45

PC	SP	X	Y	A	B	SXHI NZVC
150D	3C00	0000	0000	65	98	1001 0001

STAA $1501

PC	SP	X	Y	A	B	SXHI NZVC
150F	3C00	0000	0000	1F	98	1001 0000

LDAA #$87

PC	SP	X	Y	A	B	SXHI NZVC
1512	3C00	0000	0000	1F	98	1001 0000

SBCA #$23

PC	SP	X	Y	A	B	SXHI NZVC
1514	3C00	0000	0000	87	98	1001 1000

STAA $1500

PC	SP	X	Y	A	B	SXHI NZVC
1516	3C00	0000	0000	64	98	1001 0010

SWI

PC	SP	X	Y	A	B	SXHI NZVC
1519	3C00	0000	0000	64	98	1001 0000