CS 3468 – Project

Stuart Olsen

December 9, 2014

Implementation

```
.include "m128def.inc"

start:
        ; Set up the stack
        ldi r16, low(RAMEND)
        ldi r17, high(RAMEND)
        out SPL, r16
        out SPH, r17
        ; Clear SREG
        ldi r16, 0x00
        out SREG, r16
        ; Test
        ldi r26, 0x6B ; X = 4203
        ldi r27, 0x10
        ldi r28, 0xF6 ; Y = -10
        ldi r29, 0xFF
        call Div2
        ; Done
stop:
        rjmp stop

U_Mult2:
        ;; Truncating unsigned multiplication:
        ;; r24:r25 = r26:r27 * r28:r29
        in r16, SREG
        ; R = X0 * Y0
        mul r26, r28
        mov r24, r0
        mov r25, r1
        ; R += X1 * Y0 * 2^8
        mul r27, r28
        add r25, r0
        ; R += X0 * Y1 * 2^8
```

```
        mul r26, r29
        add r25, r0
        ; Done
        out SREG, r16
        ret

Mult2:
        ;; Truncating signed multiplication:
        ;; r24:r25 = r26:r27 * r28:r29
        in r16, SREG
        ; R = X0 * Y0
        mul r26, r28
        mov r24, r0
        mov r25, r1
        ; R += X1 * Y0 * 2^8
        muls r27, r28
        add r25, r0
        ; R += X0 * Y1 * 2^8
        muls r26, r29
        add r25, r0
        ; Done
        out SREG, r16
        ret

U_Div2:
        ;; Unsigned integer division:
        ;; r24:r25, r30:r31 = floor (r26:r27, r28:r29)
        ;;
        ;; The algorithm used is:
        ;;
        ;;   q := 0
        ;;   r := 0
        ;;   N := X
        ;;   for i = 0..15 do
        ;;     r := r << 1
        ;;     r(0) := N(15)
        ;;     N := N << 1
        ;;     q := q << 1
        ;;     if r >= Y then
        ;;       r = r - Y
        ;;       q(0) := 1
        ;;     end
        ;;   end
        ;;
        in r16, SREG
```

```
U_Div2_noarg:
        ; q := 0
        eor r24, r24
        eor r25, r25
        ; r := 0
        eor r30, r30
        eor r31, r31
        ; N := X
        movw r22:r23, r26:r27
        ; for i = 0..15 do
        eor r17, r17
        _loop:
                ; r := r << 1
                lsl r30
                rol r31
                ; r(0) := N(15)
                bst r23, 7
                bld r30, 0
                ; N := N << 1
                lsl r22
                rol r23
                ; q := q << 1
                lsl r24
                rol r25
                ; if r >= Y then
                cp r30, r28
                cpc r31, r29
                brlt _continue
                        ; r = r - Y
                        sub r30, r28
                        sbc r31, r29
                        ; q(0) := 1
                        ori r24, 1
        _continue:
                inc r17
        cpi r17, 16
        brlt _loop
        ; Done
        out SREG, r16
        ret

Div2:
        ;; Signed integer division:
        ;; r24:r25, r30:r31 = floor (r26:r27, r28:r29)
        in r16, SREG
```

```
push r26
push r27
push r28
push r29
; if Y < 0
cpi r29, 0
brge _canonicalized
        ; X := -X
        com r26
        com r27
        adiw r26:r27, 1
        ; Y := -Y
        com r28
        com r29
        adiw r28:r29, 1
_canonicalized:
; If X >= 0
cpi r27, 0
brlt _negative
        ; floor (X, Y)
        call U_Div2_noarg
        rjmp _end
_negative:
; q, r := floor (-X, Y)
com r26
com r27
adiw r26:r27, 1
call U_Div2_noarg
; q := -q
com r24
com r25
adiw r24:r25, 1
; r := -r
com r30
com r31
adiw r30:r31, 1
;; Ensure 0 <= r < Y:
cpi r31, 0
breq _end
; q := q - 1
sbiw r24:r25, 1
; r := r + Y
add r30, r28
adc r31, r29
_end:
```

```
; Done
pop r29
pop r28
pop r27
pop r26
out SREG, r16
ret
```

Screenshots

U_Mult2

R24	0x38
R25	0x7E
R26	0x78
R27	0x02
R28	0x09
R29	0x03
R30	0x00
R31	0x00

U_Div2

R24	0x06
R25	0x00
R26	0x1A
R27	0x07
R28	0x2C
R29	0x01
R30	0x12
R31	0x00

Mult2

R24	0xC8
R25	0x81
R26	0x88
R27	0xFD
R28	0x09
R29	0x03
R30	0x00
R31	0x00

Div2

R24	0x5B
R25	0xFE
R26	0x95
R27	0xEF
R28	0x0A
R29	0x00
R30	0x07
R31	0x00