Homework #5

Illya Starikov

Dude Date: November 1st, 2016

Problem #1

Letter	Ascii Hex
R	52_{16}
o	$6F_{16}$
l	$6C_{16}$
l	$6C_{16}$
a	61_{16}
$\sum$	$1FA_{16}$

Because we ignore carries, we consider FA_{16}.

$$\begin{aligned} FA_{16} &= 1111\ 1010_2 \\ \text{1's comp} &= 0000\ 0101_2 \\ \text{2's comp} &= 0000\ 0110_2 \\ &= 06_{16} \end{aligned}$$

The checksum value is 06_{16}. Performing the checksum, we get $FA_{16} + 06_{16} = 100_{16}$. Because we ignore carries, the check is correct.

Problem #2

```
        ORG 200H
STRING: DB 'Missouri S&T'
        ORG 0H

REPEAT: MOV R0, #12D                ; Size of 'Missouri S&T'
        MOV DPTR, #STRING
        ACALL BLINK
```

```
LOOP:   MOV P1, #0H                 ; P1 = output port
        CLR A
        MOVC A, @A+DPTR
        MOV P1, A                   ; P1 = string
        INC DPTR                    ; DPTR++
        ACALL DELAY

        DJNZ R0, LOOP

        ACALL BLINK
        ACALL BLINK
        ACALL DELAY
        ACALL DELAY
        SJMP REPEAT

DELAY:  MOV R7, #118D               ; 256 / 1.085 = 235.9269 * 1/2 (djnz) = 118
HERE:   DJNZ R7, HERE
        RET

BLINK:  MOV P0, #0H                 ; Make output port
        CLR P0.1                    ; Low
        SETB P0.1                   ; To High
        CLR P0.1                    ; To low
        RET

        END
```

Problem #3

```
        ORG 0H

        /*
        PORTS
        R0: Loop counter
        R1: Holds hundreds, tens, ones
        R2: Hold the additive value
        */

        MOV P1, #0FFH               ; P1 = Input Port
        MOV P2, #0H                 ; P2 = Output Port
        MOV P1, #32H
```

```
REPEAT: MOV R0, #3D                ; Number to read in
        MOV R1, #100D
        MOV R2, #0D

        CLR A
        CLR C

LOOP:   MOV A, P1
        ANL A, #0FH                ; By and-ing with 0F, upper bits are cleared.
                                   ; This way, goes from ascii to a hex number
        MOV B, R1
        MUL AB                     ; Multiply by hundreds, tens, one+

        ADD A, R2                  ; Add to the values we have received so far
        MOV R2, A                  ; R2 = Sum value

        ACALL SWAPAB
        JC BIG                     ; if this additive value sum is > 255, Carry
        JNZ BIG                    ; if there was something in b, Carry

        MOV A, R1                  ; A = Tens, hundred, ones
        MOV B, #10D
        DIV AB                     ; So now, we have values for tens, ones
        MOV R1, A

        ACALL DELAY1
        DJNZ R0, LOOP

        MOV P2, R2

        SJMP SKIP
BIG:    ACALL DELAY2
SKIP:   LJMP REPEAT

        ACALL REPEAT


DELAY1: MOV R7, #118D              ; 256 / 1.085 = 235.9269 * 1/2 (djnz) = 118
HERE1:  DJNZ R7, HERE1
        RET
```

```
DELAY2: MOV R6, #4D
HERE2:  ACALL DELAY1                ; 1.024 ms = 1024 us = 4*Delay1
        DJNZ R6, HERE2
        RET


SWAPAB: MOV R5, A
        MOV A, B
        MOV B, R5
        RET


        END
```

Problem #4

```
        ORG 250H
MYDATA: DB 3, 9, 6, 9, 7, 6, 4, 2, 8
        ORG 0H

        ACALL R2R
        ACALL ADDV
        ACALL AVG

        ACALL DONE

R2R:    MOV R0, #30H
        MOV DPTR, #MYDATA
        MOV R1, #9D                 ; Size of 'MYDATA'

LOOP1:  CLR A
        MOVC A, @A+DPTR
        MOV @R0, A

        INC R0
        INC DPTR
        DJNZ R1, LOOP1

        RET
```

```
ADDV:   MOV R0, #30H                ; Where values are stored
        MOV R1, #9D                 ; Size of 'MYDATA'
        CLR A
LOOP2:  ADD A, @R0
        INC R0

        DJNZ R1, LOOP2

        MOV 70H, A
        RET


AVG:    MOV R0, #30H                ; Where values are stored
        MOV R1, #0D                 ; Counter
        MOV R2, #0D                 ; Sum

        CLR A
        /*
        So.. not entirely sure how to do this part, so here's my guess..

        I don't write to ROM anywhere past 30H, so it should theoretically be 0
        So keep reading and adding until I hit an 0 in RAM
        */
LOOP3:  MOV A, @R0
        JZ LDONE

        ADD A, R2
        MOV R2, A

        INC R0
        INC R1
        SJMP LOOP3

LDONE:  MOV B, R1
        MOV A, R2
        DIV AB

        MOV R7, A

        RET

DONE:   NOP
```

```
        END
```

Problem #5

```
        ORG 0

        /*
        R0 = x

        R1 = 2x^2
        R2 = 3x
        R3 = 1
        */

        MOV P1, #0FFH               ; P1 = input port
        MOV P2, #0H                 ; P2 = output port

        MOV R0, P2                  ; R0 = P2

        CLR C                       ; CY = 0
        CJNE R0, #4D, NEXT
NEXT:   JC SKIP                     ; if CY = 0, outside of range (by CJNE)
        MOV P2, #0H                 ; R2 = 0, if outside of 0 <= x <= 3
        ACALL DONE                  ; Finish

SKIP:    /* 2x^2 */

        MOV A, R0                   ; A = x
        MOV B, R0                   ; B = x
        MUL AB                      ; A = x^2, but the lower value. However,
                                    ; x will always be in the lower bit, as in A
        MOV B, #2D
        MUL AB                      ; A = 2x^2
        MOV R1, A                   ; R1 = 2x^2

        /* 3x */
        MOV A, R0                   ; A = x
        MOV B, #3D                  ; B = 3
        MUL AB                      ; A = 3x

        MOV R2, A                   ; R2 = 3x
```

```
        /* 1
         I know this is kinda wasting MC, but
         it's easier to manipulate the equation this way
         */
        MOV R3, #1D    ; R3 = 1

        MOV A, R1
        ADD A, R2
        ADD A, R3                   ; A = 2x^2 + 3x + 1

        MOV P2, A                   ; P2 = 2x^2 + 3x + 1

DONE:   NOP
        END
```