ENGI 4862
Microprocessors

Assignment 4

Due Friday, 14 July 2017, 4:00 PM

1 Subroutine Delay Calculation [15]

(a) Find the time delay for the delay subroutine shown below if the system has an AVR with a frequency of 4 MHz. [**12 marks**]

```
DELAY:
        LDI R16, 250
OUTER:
        LDI     R17, 155
INNER:
        DEC R17
        BRNE INNER
        DEC R16
        BRNE OUTER
        RET
```

(b) Based on part (a), how much of a total time delay will executing the instruction `CALL DELAY` before executing whatever instruction comes after it? Would the RCALL subroutine be any different? [**3 marks**]

2 Timer Programming [30]

(a) Assume that XTAL = 10 MHz. Find the TCNT0 value needed to generate a time delay of $12\mu s$ using Normal Mode, no prescaler mode. Write the assembly code to set this up. [**5 marks**]

(b) Assume that XTAL = 2 MHz. Find the OCR0 value needed to generate a time delay of 0.05 ms using CTC mode, no prescaler mode. Write the assembly code to set this up. [**5 marks**]

(c) Program Timer0 with normal mode and prescaler of 8 to generate a square wave of 2 KHz on the PORTB.7 bit. Assume that XTAL = 8 MHz. [**10 marks**]

(d) Program the ATmega32 Timer1 to be an event counter (triggered by event falling edges). Use simple CTC Mode, and display the binary count on PORTC and PORTD (PORTC for high byte and PORTD for low byte) continuously. If the events tally to 25,000 this event counter will be restarted. [**10 marks**]

3 Instruction Encoding [15]

Manually convert the following instructions to machine code by referring to the Instruction Set Manual. Give your answers in hexadecimal.

```
.INCLUDE "M32DEF.INC"

.ORG 0
LDI R16, HIGH(RAMEND)
OUT SPH, R16
LDI R16, LOW(RAMEND)
OUT SPL, R16
LDI R16, 0xFF
OUT DDRA, R16
CLR R16
OUT DDRB, R16

BACK:
        IN R24, PINB
        CALL DELAY
        OUT PORTA, R24
        CALL DELAY
        INC R16 ; changed from ADDI R16, 1
        CPI R16, 100
        BRNE BACK

.ORG 0X60
DELAY:
        PUSH R24
        LDI R24, 0XCE
AGAIN:
        DEC R24
        BRNE AGAIN
        POP R24
        RET
```

4 String Copying [20]

Write a program that scans an ASCII string located at SOURCE_STR (in the code memory) to check that it contains at least one (upper or lowercase) ASCII text character (as opposed to just numbers, punctuation, other symbols, etc.) *and* its length is more than 0 and less than 256. The end of the string is indicated by a 0 byte.

If the source string passes the check, copy it to DEST_STR in the data memory (including the terminating 0); otherwise store just a 0 to DEST_STR.

```
.CSEG
.ORG 0x250
SOURCE_STR:
  .DB "This is an example string that *should* be copied.", 0

.DSEG
.ORG 0x100
DEST_STR: .BYTE 256
```

5 ASCII String Addition [20]

Write an ATmega32-compatible program that adds two 10-digit ASCII numbers (which are located in code ROM), that is, DATA1 + DATA2. The result should be saved to RESULT in ASCII located in data RAM (accouting for the possibility of carry-out). They can be defined as follows:

```
.CSEG
.ORG 0x100
DATA1: .DB "2468101214"
DATA2: .DB "1357911130"

.DSEG
.ORG 0x100
RESULT: .BYTE 11
```

Note that there are many ways you can approach solving this problem:

- Convert the ASCII representation to a 36-bit binary number and then add the two 5-byte representations.
- Add the corresponding ASCII digits 10 times, and write code to handling carry out and conversions to ensure the result looks correct.
- Convert to a BCD representation (or packed BCD) and adding that way, then converting back.

- Many other options…

Once you pick an approach, writing some pseudocode to help you figure out your algorithm is probably a good idea before you start the assembly coding.