COE 306, Term 171

Introduction to Embedded Systems

Assignment# 1 Solution
Due date: Sunday, Oct. 8, 2017

Q.1. Look for a consumer product, e.g. a toy or a small home appliance that incorporates an embedded system. It will be good if you can find a locally available product, and include pictures showing its computer components.

Note: You may need to disassemble the product to be able to verify that it does incorporate an embedded system. If this is the case, please exercise caution and follow safety procedures to avoid any injuries. If unsure about the safety of an action, consult lecture or lab instructors.

(a) What feature of the product suggested to you that it may incorporate an embedded system?

(b) What is the model number of the microcontroller chip used in this product?

(c) What is the architecture of this microcontroller?

(d) What are the main specifications of this microcontroller? On-chip memory, integrated peripheral devices, maximum frequency, power rating (volts, amperes, watts), any other notable specifications.

(e) Provide the URL of the microcontroller datasheet.

(f) Include a picture of the product in its original condition, and another picture of its main board, showing the main microcontroller.

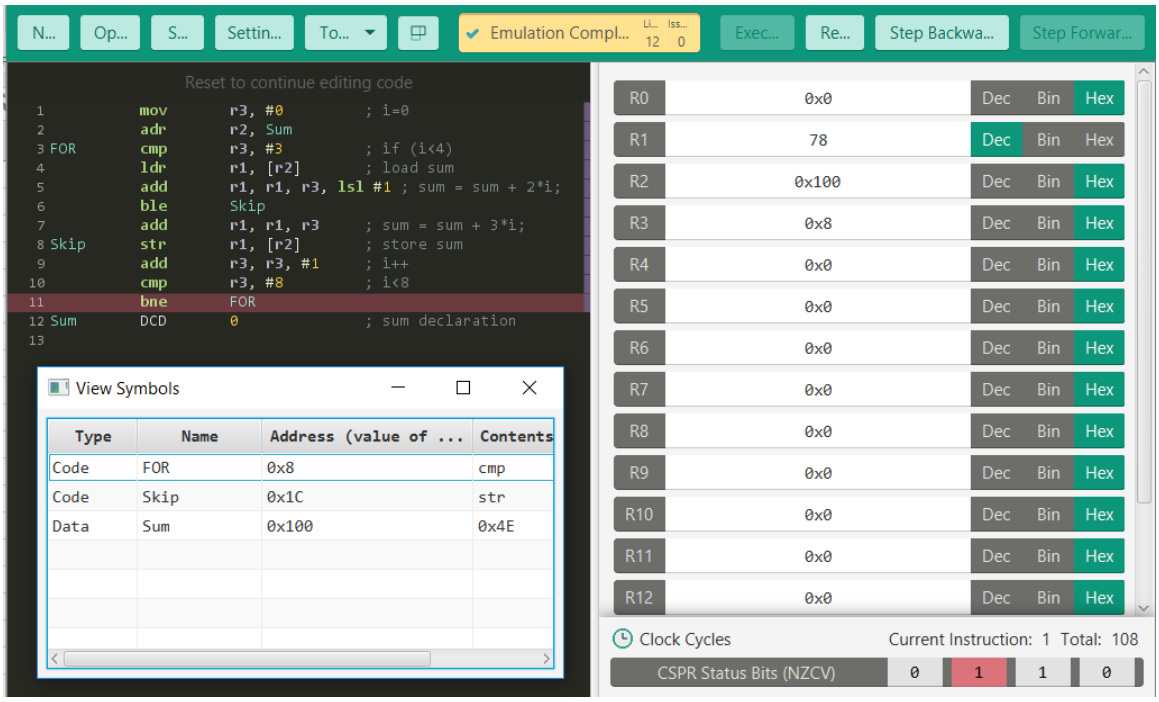
Q.2. Consider the C code given below:

```
volatile static int sum = 0; // assume sum is a memory variable
for (int i=0; i<8; i++){
      if (i<4)
            sum = sum + 2*i;
      else
            sum = sum + 3*i;
}
```

(a) Write the given C code in ARM assembly without the use of conditional instructions except branch. Simulate your program using VisUAL ARM emulator and include a snapshot to show that your program produced the correct result of sum.

```
        mov     r3, #0          ; i=0
        adr     r2, Sum
FOR     cmp     r3, # 3         ; if (i<4)
        ldr     r1, [r2]        ; load sum
        add     r1, r1, r3, lsl#1 ; sum = sum + 2*i;
```

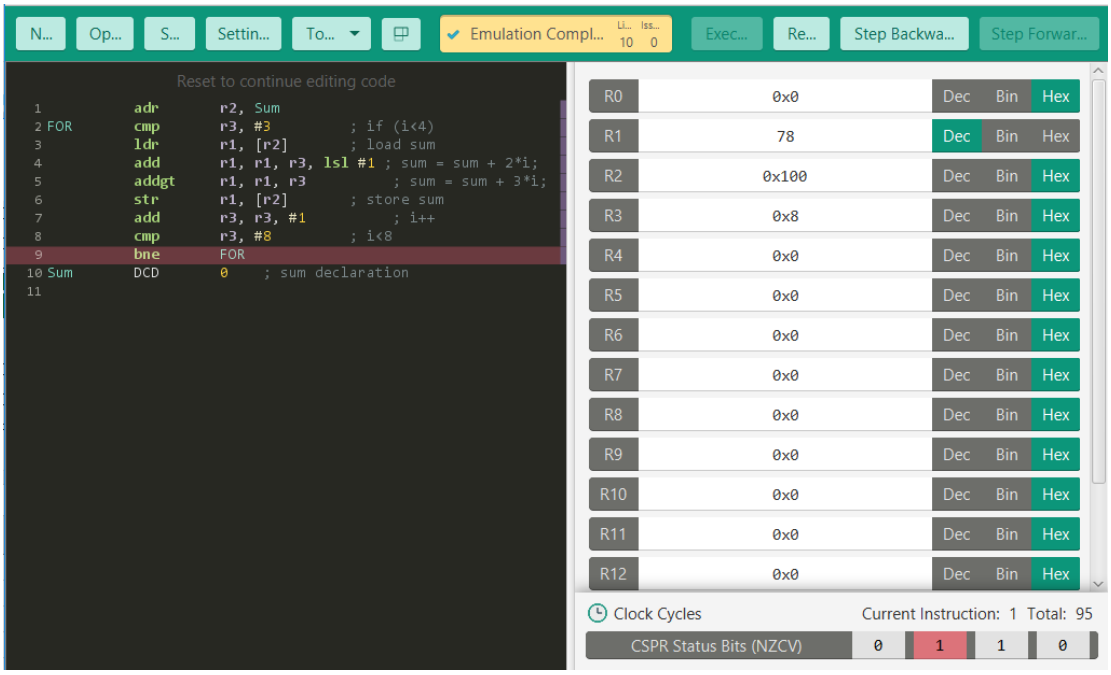
```
          ble       Skip
          add       r1, r1, r3      ; sum = sum + 3*i;
Skip      str       r1, [r2]        ; store sum
          add       r3, r3, #1      ; i++
          cmp       r3, #8          ; i<8
          bne       FOR
Sum       DCD       0               ; sum declaration
```



(b) Write the given C code in ARM assembly with the use of conditional instructions. Simulate your program using VisUAL ARM emulator and include a snapshot to show that your program produced the correct result of sum.

```
          mov       r3, #0          ; i=0
          adr       r2, Sum
FOR       cmp       r3, #3          ; if (i<4)
          ldr       r1, [r2]        ; load sum
          add       r1, r1, r3, lsl #1 ; sum = sum + 2*i;
          addgt     r1, r1, r3      ; sum = sum + 3*i;
          str       r1, [r2]        ; store sum
          add       r3, r3, #1      ; i++
          cmp       r3, #8          ; i<8
```

```
        bne     FOR
Sum     DCD     0       ; sum declaration
```



(c) Compare your assembly codes in part (a) and part (b) in terms of number of instructions in each code and the number of instructions executed in each code.

As one can see, using conditional instructions saves one instruction in the code size and one less instruction to be executed in every loop iteration. More importantly a conditional branch instruction is avoided which could result in flushing pipeline when this is branch misprediction.

(d) Compile your C program to ARM assembly code and compare your handwritten assembly program in part (b) with the compiler-generated assembly code using LPCXpresso IDE. Explain any differences between the two assembly programs.

The relevant code generated by the compiler is given below. As can be seen our code in (b) is more efficient that the code generated by the compiler. In addition, the way data is declared and the way address of sum is loaded are different.

```
mov  r3, #0
ldr  r2, .L6
.L4:
```

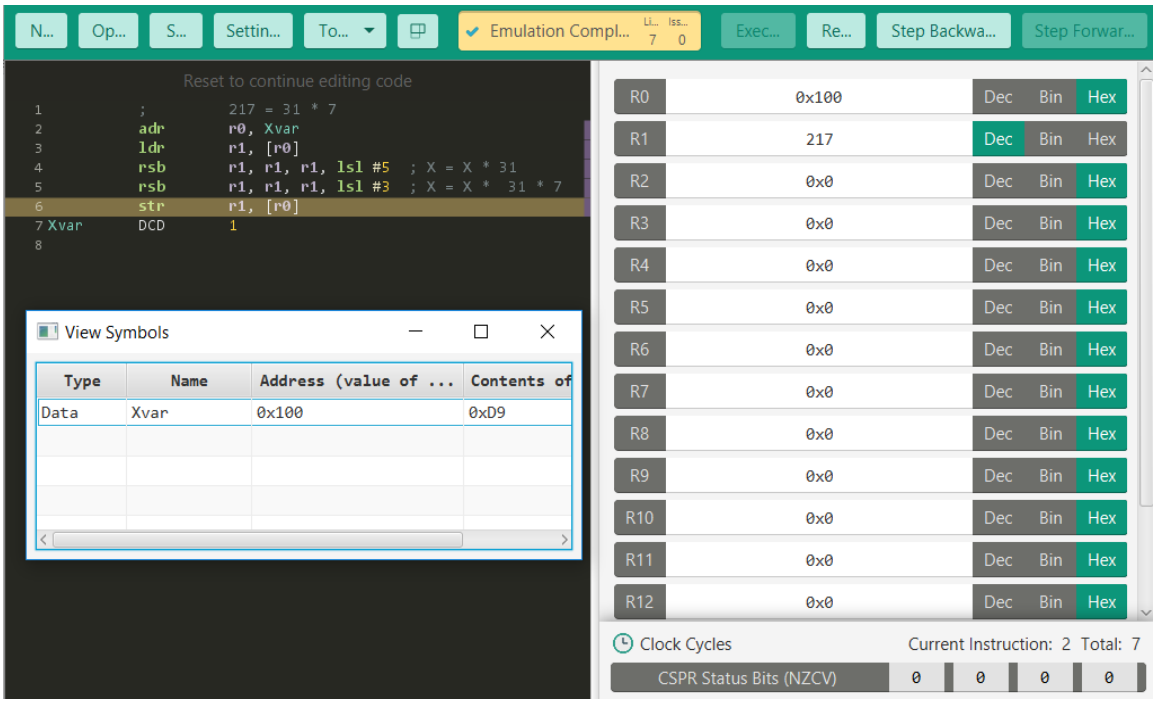
```
cmp  r3, #3
ldrle      r1, [r2]
addle     r1, r1, r3, lsl #1
ldrgt     r0, [r2]
addgt     r1, r3, r3, lsl #1
addgt     r1, r1, r0
str  r1, [r2]
add  r3, r3, #1
cmp  r3, #8
bne  .L4
.L6:
.word     .LANCHOR0
.LFE0:
.section   .bss.sum.3686,"aw",%nobits
.align    2
.set .LANCHOR0,. + 0
sum.3686:
.space    4
```

Q.3. Consider the C code given below:

```
volatile static int x; // assume x is a memory variable
x = x * 217;
```

(a) Write the given C code in ARM assembly without the use of multiplication instructions. Simulate your program using VisUAL ARM emulator and include a snapshot to show that your program produced the correct result of sum.

```
; 217 = 31 * 7
          adr        r0, Xvar
          ldr        r1, [r0]
          rsb        r1, r1, r1, lsl #5    ; X = X * 31
          rsb        r1, r1, r1, lsl #3    ; X = X *  31 * 7
          str        r1, [r0]
Xvar      DCD        1
```



(b) Compile your C program to ARM assembly code and compare your handwritten assembly program in part (b) with the compiler-generated assembly code using LPCXpresso IDE. Explain any differences between the two assembly programs.

The relevant code generated by the compiler is given below, which is similar to our code with the difference of the order of multiplication of 31 and 7.

```
ldr  r2, .L2
ldr  r3, [r2]
rsb  r3, r3, r3, lsl #3
rsb  r3, r3, r3, lsl #5
str  r3, [r2]
.L2:
.word     .LANCHOR0
.LFE0:
.section   .bss.x.3686,"aw",%nobits
.align    2
.set .LANCHOR0,. + 0
x.3686:
.space    4
```