

COE 306, Term 161

Introduction to Embedded Systems

Assignment# 1 Solution

Due date: Thursday, Oct. 13, 2016

- 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<10; i++){
    if (i<5)
        sum = sum + i;
    else
        sum = sum + 2*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, #4          ; if (i<5)
         ldr      r1, [r2]       ; load sum
         add      r1, r1, r3     ; sum = sum + i;
         ble     Skip
         add      r1, r1, r3     ; sum = sum + 2*i;
Skip     str      r1, [r2]       ; store sum
         add      r3, r3, #1     ; i++
         cmp     r3, #10        ; i<10
         bne     FOR
Sum      DCD     0              ; sum declaration

```

The screenshot shows the VisUAL ARM emulator interface. On the left, the assembly code is displayed with line numbers 1 through 13. The code includes instructions for moving values, loading and storing data, conditional branching, and arithmetic operations. On the right, a register window displays the state of registers R0 through R8. Register R0 contains 0x0, R1 contains 80, R2 contains 0x100, R3 contains 10, and registers R4 through R8 all contain 0x0. The register values are shown in decimal, binary, and hexadecimal formats.

(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, #4          ; if (i<5)
         ldr     r1, [r2]       ; load sum
         addle  r1, r1, r3     ; sum = sum + i;
         addgt  r1, r1, r3, lsl #1 ; sum = sum + 2*i;
         str     r1, [r2]       ; store sum
         add     r3, r3, #1     ; i++
         cmp    r3, #10        ; i<10

```

```

                bne     FOR
Sum             DCD     0                ; sum declaration

```

The screenshot shows an ARM assembly editor with the following code:

```

1      mov     r3, #0      ; i=0
2      adr     r2, Sum
3  FOR  cmp     r3, #4      ; if (i<
4      ldr     r1, [r2]    ; load s
5      addle   r1, r1, r3  ; sum =
6      addgt   r1, r1, r3, lsl #1 ; su
7      str     r1, [r2]    ; store
8      add     r3, r3, #1  ; i++
9      cmp     r3, #10     ; i<10
10     bne     FOR
11  Sum  DCD     0         ; sum de
12

```

On the right, a register window shows the state of registers R0 through R8:

Register	Value	Dec	Bin	Hex
R0	0x0			
R1	80			
R2	0x100			
R3	10			
R4	0x0			
R5	0x0			
R6	0x0			
R7	0x0			
R8	0x0			

(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, which is identical to our code written for part (b), except for the way data is declared and the address of sum is loaded:

```

mov r3, #0
ldr r2, .L6
.L4:
cmp r3, #4
ldr r1, [r2]
addle r1, r1, r3
addgt r1, r1, r3, lsl #1
str r1, [r2]
add r3, r3, #1
cmp r3, #10

```

```

bne .L4
.L6:
.word .LANCHOR0
.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=1; // assume x is a memory variable
x = x * 64449;

```

(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.

; 64449 = 1023 * 63

```

adr    r0, Xvar
ldr    r1, [r0]
rsb    r1, r1, r1, lsl #6    ; X = X * 63
rsb    r1, r1, r1, lsl #10   ; X = X * 63 * 1023
str    r1, [r0]
Xvar   DCD    1

```

The screenshot shows the VisUAL ARM emulator interface. On the left, the assembly code is displayed with line numbers 1 through 7. Line 5 is highlighted. On the right, the register window shows the state of registers R0 through R4. R0 contains 0x100, R1 contains 64449, and R2 through R4 contain 0x0.

Register	Value	Dec	Bin	Hex
R0	0x100			Hex
R1	64449	Dec	Bin	Hex
R2	0x0	Dec	Bin	Hex
R3	0x0	Dec	Bin	Hex
R4	0x0	Dec	Bin	Hex

(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 identical to our code, except for the way data is declared and the way the address of X is loaded:

```
ldr r2, .L2
ldr r3, [r2]
rsb r3, r3, r3, lsl #6
rsb r3, r3, r3, lsl #10
str r3, [r2]
.L2:
.word .LANCHOR0
.LFE0:
.section .data.x.3686,"aw",%progbits
.align 2
.set .LANCHOR0,. + 0
x.3686:
.word 1
```