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;
}</pre>
```

(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, l	sl#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

V Op	S S	Settin To 🔻	Emulation (Compl ^{Li} Iss 12 0	Exec Re Ste	ep Backwa	Step Forw	ar
				RO	0×0	Dec	Bin He	x
1 2		r3, #0 ; i=∣ r2, Sum			0,10			~
∠ 3 FOR		r2, Sum r3, #3 ; if	(i<4)	– R1	78	Dec	Bin He	x
	ldr i	r1, [r2] ; lo			0.400	Dee		
5 6		r1, r1, r3, lsl #1		R2	0×100	Dec	Bin He	x
ь 7		Skip r1,r1,r3 ; su	m = sum + 3*i;	R3	0x8	Dec	Bin He	x
8 Skip								=
		r3, r3, #1 ; i+-		R4	0×0	Dec	Bin He	x
0 1		r3,#8 ;i< FOR	8				n: 11	
2 Sum			m declaration	R5	0×0	Dec	Bin He	x
				R6	0×0	Dec	Bin He	×
View S	ymbols	-	- 🗆 X	R7	0×0	Dec	Bin He	x
Туре	Name	Address (value	of Contents	R8	0x0	Dec	Bin He	x
Type Code	Name FOR	Address (value 0x8	of Contents	R8 R9	0×0 0×0		Bin He Bin He	-
		•		R9	0×0	Dec	Bin He	x
Code	FOR	0x8	стр			Dec		x
Code Code	FOR Skip	0x8 0x1C	cmp str	R9	0×0	Dec Dec	Bin He	ex ex
Code Code	FOR Skip	0x8 0x1C	cmp str	R9 R10	0x0 0x0	Dec Dec Dec	Bin He	ex ex ex
Code Code	FOR Skip	0x8 0x1C	cmp str	R9 R10 R11	0x0 0x0 0x0 0x0	Dec Dec Dec	Bin Her Bin Her Bin Her Bin Her	

(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	стр	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

N (Op S	Settin	То 🔻	₽	 Emulation Co 	ompl ^{Li} Iss 10 0	Exec Re	Step Backwa	Step	Forwar
1	Res a dr	et to contine r2, Sum	ue editing co	de		RO	0×0	Dec	Bin	Hex
	cmp 1dr	r3, #3				R1	78	Dec	Bin	Hex
	add addgt	r1, r1,	r3			R2	0x100	Dec	Bin	Hex
	str add	r3, r3,	; s #1 .			R3	0x8	Dec	Bin	Hex
8 9 10 Sum	cmp bne DCD	r3, #8 FOR	; i um declara			R4	0×0	Dec	Bin	Hex
10 Sum 11	DCD	0;5	um deciara	LTON		R5	0×0	Dec	Bin	Hex
						R6	0×0	Dec	Bin	Hex
						R7	0×0	Dec	Bin	Hex
						R8	0×0	Dec	Bin	Hex
						R9	0×0	Dec	Bin	Hex
						R10	0×0	Dec	Bin	Hex
						R11	0×0	Dec	Bin	Hex
						R12	0×0	Dec	Bin	Hex
						Clock Cyc	cles	Current Instruct	ion: 1	Total: 95
						CSPR S	Status Bits (NZCV)	0 1	1	0

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

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.

N Op	S	Settin	То 🔻	₽	 Emulation C 	ompl ^{Li} Iss 7 0	Exec Re	Step Backwa	Step	Forwar
1	Res	et to continue 217 = 31		de		RO	0x100	Dec	Bin	Hex
	adr 1dr	r0, Xvar r1, [r0]				R1	217	Dec	Bin	Hex
	rsb rsb	r1, r1, r	1, lsl #5 1, lsl #3			R2	0×0	Dec	Bin	Hex
6 7 Xvar	str DCD	r1, [r0] 1				R3	0×0	Dec	Bin	Hex
						R4	0×0	Dec	Bin	Hex
						R5	0×0	Dec	Bin	Hex
View S	ymbols			-		R6	0×0	Dec	Bin	Hex
Туре	Name		ss (value	of	Contents of	R7	0×0	Dec	Bin	Hex
Data	Xvar	0x100			0xD9	R8	0×0	Dec	Bin	Hex
						R9	0×0	Dec	Bin	Hex
						R10	0×0	Dec	Bin	Hex
<					>	R11	0×0	Dec	Bin	Hex
						R12	0×0	Dec	Bin	Hex
						Clock Cyc	les	Current Instruc	tion: 2	Total:
						CSPR S	itatus Bits (NZCV)	0 0	0	0

(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
          .LANCHORØ
.LFE0:
.section
          .bss.x.3686,"aw",%nobits
          2
.align
.set .LANCHOR0,. + 0
x.3686:
          4
.space
```