Name ____

ELC 411 – Embedded Systems
Fall 2015
Larry Pearlstein

Mid-term exam

2 points

1) Give an example of a device containing an embedded system

6 points

2) Address spaces

a. How many address bits would be required to represent a 256 MByte address space?

28 bits $= \log_2(256 \times 1024 \times 1024)$

b. How large is the address space generated by a 31-bit address?

2 GBytes $= 2^{31}$

8 points

3) Complete the following arithmetic operations in two's complement representation. What are the values of the carry flag (C) and the overflow flag (V)? **Assume a 5-bit system.**

a. 14 – 15 (show me the binary subtraction)

N = 5

$$\begin{array}{rccccc} & & 1 & 10 & 10 & 10 \\ & 0 & \cancel{1} & \cancel{1} & \cancel{1} & 0 \\ - & 0 & 1 & 1 & 1 & 1 \\ \hline B & 1 & 1 & 1 & 1 & 1 \end{array}$$

$\boxed{N=1 \quad Z=0}$

C = 0

14 – 15 = –1 $\underline{\underline{V=0}}$

b. -8 + (-16) (show me the binary addition)

N=5 RANGE SIGNED
–16 to +15

$$\begin{array}{r} -8 \\ +-16 \\ \hline \underline{\underline{-24}} \end{array}$$

$\boxed{V=1}$

$$\begin{array}{rccccc} & 1 & 1 & 0 & 0 & 0 \\ + & 1 & 0 & 0 & 0 & 0 \\ \hline C & \boxed{0} & \boxed{1} & \boxed{0} & \boxed{0} & \boxed{0} \end{array}$$

= +8

$\boxed{C=1}$

$\boxed{N=0}$ $\boxed{Z=0}$

Name ______________________________

8 points

4) Write the following procedure (use the opposite blank page) in 'C' code

```
// Copy the null-terminated byte string starting at address
// 'src' to a memory array starting at address 'dst'.  The
// terminating '\0' character is also copied.
void strcpy(char *dst, char *src);
```

8 points

5) ARMv7 (used in Cortex-M3 CPU core)

a. What is the broad classification of the Cortex-M3 memory access architecture (circle letter of the correct answer):

i. von Neumann

[✓ ii.] (modified) Harvard

b. What are the sizes of coded machine instructions in the ARMv7 ISA, in bits?

16 & 32 BITS

c. How many fully general purpose registers are there in the ARMv7 ISA?

13

d. What are the names and functions of the registers numbered r13 and r15 in the ARMv7 ISA

R13= stack pointer

R14=link register

R15 = program ctr ✓

5 points

6) Refer to the byte array shown on the last page. Assuming the memory is accessed using a big-endian interpretation, what 32-bit word value would be read at address 0x104?

FACE B00C

8 points

7) Suppose r0 = 0xFF00FF00 and r1 = 0xAA55AA55, find the result of the following operations:

a. EOR r2, r0, r1

55 55 55 55

b. ORR r2, r0, r1

FF 55 FF 55 FF 55

Name ____

```
void
strcpy (char *dst, char *src)
{
      while (*src)  {
          *dst = *src;
          dst++;
          src++;
      }
      *dst = '\0';
}
```

Name ____

7 points

8) The ARMv7 ISA defines the adds instruction, which adds the 32-bit values from two source operands, and stores the 32-bit result in a destination operand.

Consider the statement:

adds r4, r2, r3

where the values of r2 and r3 are 0xFFFF FFF3 and 0x0000 000C respectively.

a. What does the character 's' signify in the instruction '**adds**'? set flags
b. Is the 'adds' instruction used for adding signed operands, unsigned operands, or both? BOTH
c. What **decimal** value is represented by r2, if a signed interpretation is used? −13
d. What **decimal** value is represented by r3, if an unsigned interpretation is used? 12
e. What **decimal** value is represented by r3, if a signed interpretation is used? 12
f. What is the value stored in register r4, represented in **hexadecimal**? FFFF FFFF
g. What are the states of the ARM NZCV bits after the operation?

N = 1   Z = 0   C = 0   V = 0

12 points

9) Suppose r0 = 0x0100, sp=0x0110 and the memory has been initialized as shown on the last page.

Fill in the values of r0, r1, r2, sp that would be produced **after** each of instructions in the following sequence is executed, assuming little endian format.

INSTR	r0	r1	r2	sp
INITIAL	0x100	x	x	0x110
LDR r1, [r0]	0x100	00000110	x	0x110
LDR r2, [r0, #8]!	0x108	"	DDCCBBAA	"
LDR r2, [r0], #4	0x10C	"	DDCCBBAA	↓
STR r2, [r1], #8	"	118	↓	↓
PUSH {r0}	"	↓	↓	0x 10C
POP {r1}	"	0x10C	↓	0x 110

Name ____

Name ______________________________

7 points

10) Convert the following 'C' procedure to assembly language:

```
void add_arrays( int *arr1,int *arr2,int *arr3)
{
     int i;

     for (i = 0; i < 100; ++i)
     {
          arr3[i] = arr1[i] + arr2[i];
     }
}
```

assuming that the pointers arr1, arr2 and arr3 are passed in registers r0, r1, and r2. You can use any of the other general purpose registers, as you see fit.

4 points

11) ARMv7 ISA

a. What assembly language instruction is used to call a subroutine?

BL

b. How does one return from a subroutine in ARMv7 assembly language?

BX LR

12 points

12) Refer the assembly language program (right column) on the last sheet. When the PC is 0x100 the stack pointer (SP) is 0x20008000. ← after instr @ 0x100

a. Trace the code and fill in the PC, LR, SP and all Stack Operations, and list these in a three column table on the facing sheet for the first 9 instructions that are executed. The first entry has already been filled in for you.

b. Show the value of the LR and the entire stack contents, at the end of your instruction trace.

9 points

13) Convert the following constants to 8-bit hexadecimal values, using signed representations

a.	127	7F
b.	-128	8ø
c.	-16	Fø

4 points

14) How does a typical 'C' compiler accomplish passing the first four parameters to a procedure, in terms of the ARM assembly language?

r0, r1, r2, r3

Name ____

```
      mov  r3, #0       // i=0
loop: cmp  r3, #100
      bge  done
      ldr  r4,[r0],#4
      ldr  r5,[r1],#4             // arr1[i]+arr2[i]
      add  r5,r5,r4
      str  r5,[r2],#4
      add  r3,r3,#1
      b    loop
done: bx lr
```

20008008

Program Counter	Link Register (LR)	Stack Pointer (SP)	Stack Op (push/pop & value)
0x100	0x010	0x20008000	push 0x010
0x104			
108			
10C			
110			
114	118		
124		20007FFC	PUSH 118
128	12C		
154			

12C

Name ____

Name ____

Address	Data
0x011F	0xCA
0x011E	0xFE
0x011D	0xF0
0x011C	0x0D
0x011B	0xAB
0x011A	0xAD
0x0119	0xBE
0x0118	0xEF
0x0117	0x50
0x0116	0x50
0x0115	0xFE
0x0114	0xDA
0x0113	0xC0
0x0112	0x0C
0x0111	0x1E
0x0110	0x20
0x010F	0x00
0x010E	0x11
0x010D	0x22
0x010C	0x33
0x010B	0xDD
0x010A	0xCC
0x0109	0xBB
0x0108	0xAA
0x0107	0xOC
0x0106	0xBO
0x0105	0xCE
0x0104	0xFA
0x0103	0x00
0x0102	0x00
0x0101	0x01
0x0100	0x10

Address	Label	Instruction
100	main:	push {lr}
104		mov r0, #5
108		mov r1, #4
10C		mov r2, #3
110		mov r3, #2
114		bl sop
118		mul r0, r0, r0
11C		pop {lr}
120		bx lr
124	sop:	push {lr}
128		bl product
12C		push {r0}
130		push {r2}
134		push {r3}
138		pop {r0}
13C		pop {r1}
140		pop {r2}
144		bl product
148		add r0, r0, r2
14C		pop {lr}
150		bx lr
154	product:	mul r0, r0, r1
158		bx lr