The University of Alabama in Huntsville
Electrical and Computer Engineering Department
CPE 221 01
Test 1 Solution
Fall 2019

This test is closed book, closed notes. You may not use a calculator. You should have the 6 page ARM Instruction Reference. *You must show your work to receive full credit.*

1. (2 points) The range of numbers that can be represented in n-bit signed 2's complement is _-2^n-1_ through _$2^{n-1} - 1$_.

2. (1 point) A _**combinational**_ logic circuit is one whose outputs depend only on its current inputs.

3. (1 point) _**False**_ (True or False) x OR x' = 0.

4. (1 point) DCW defines a constant that represents _**2**_ bytes in memory.

5. (10 points) Convert decimal +243 and -205 to binary, using signed-2's complement representation and enough digits to accommodate the numbers.

```
/2       0   1            /2       0   1
/2       1   1            /2       1   1
/2       3   1            /2       3   0
/2       7   1            /2       6   0
/2      15   0            /2      12   1
/2      30   0            /2      25   1
/2      60   1            /2      51   0
/2     121   1            /2     102   1
/2     243                /2     205

243 = 0 1111 0011
205 = 0 1100 1101, -205 = 1 0011 0011
```

6. (3 points) What is the decimal equivalent of 11010_3 (assume positional notation and unsigned integer formats)?
$1 \times 3^4 + 1 \times 3^3 + 0 \times 3^2 + 1 \times 3^1 + 0 \times 3^0 = 81 + 27 + 0 + 3 + 0 = 111$

7. (12 points) If `r1 = 0x000F 0FFF` and `r2` = 20, what is the value of `r0` after each of the following instructions has been executed? Assume that each instruction uses the same data.

```
(a)   ADD r0, r1, r1, LSL #7

r1                0000 0000 0000 1111 0000 1111 1111 1111
r1 LSL #7       + 0000 0111 1000 0111 1111 1111 1000 0000
r0                0000 0111 1001 0111 0000 1111 0111 1111 = 0x0797 0F7F
C               0 0000 0000 0001 1111 1111 1111 1000 000
(b)   ADD r0, r1, r1, ROR #8

r1                0000 0000 0000 1111 0000 1111 1111 1111
r1 ROR #8       + 1111 1111 0000 0000 0000 1111 0000 1111
r0                1111 1111 0000 1111 0001 1111 0000 1110 = 0xFF0F 1F0E
C               0 0000 0000 0000 0000 0001 1111 1111 111
```

(c) ADD r0, r1, r1, LSR r2

```
r1                  0000 0000 0000 1111 0000 1111 1111 1111
r1 LSL 20(r2)     + 0000 0000 0000 0000 0000 0000 0000 0000
r0                  0000 0000 0000 1111 0000 1111 1111 1111 = 0x000F 0FFF
C                 0 0000 0000 0000 0000 0000 0000 0000 000
```

8. (10 points) For each of the following operations on 6 bit signed numbers, calculate the values of the C, Z, V, and N flags

(a) 111010 - 001001

```
-001001 = 110111
        + 111010
S         110001
C        111110
CZVN      = 1001
```

(b) 011011 + 001010

```
011011
001010
100101
011010
CZVN = 0011
```

9. (15 points) For each of the following cases, 1) Explain the effect of each of the following instructions using register transfer notation. 2) Give the value in r2 after each instruction executes. 3) Give the value of the effective address.
Assume that r2 contains the initial value 0x0F00 ED10 and that r0 contains 0xFF9F 5400.
Use these initial values for each instruction individually.

(a) LDR r1, [r2]

Register Transfer r1 ← M[r2]
r2 **0x0F00 ED10**
Effective Address **0x0F00 ED10**

(b) STR r1, [r2, #2_1101]

Register Transfer r1 ← M[r2 + 13]
r2 **0x0F00 ED10**
Effective Address **0x0F00 ED1D**

(c) LDR r1, [r2, #0xEE]!

Register Transfer r2← r2 + 0xEE, r1 ← M[r2]
r2 **0x0F00 EDFE**
Effective Address **0x0F00 EDFE**

(d) STR r1, [r2], #4

Register Transfer M[r2] ← r1, r2 ← r2 + 4
r2 **0x0F00 ED14**
Effective Address **0x0F00 ED10**

(e) LDR r1, [r2, r0, ASR #4]

Register Transfer r1 ← M[r2 + r0 >> 4]
r2 **0x0F00 ED10**
Effective Address **0x0EFA E250**

```
r2              0000 1111 0000 0000 1110 1101 0001 0000
r0 ASR #4       1111 1111 1111 1001 1111 0101 0100 0000
EA              0000 1110 1111 1010 1110 0010 0101 0000
C             1 1111 1110 0000 0011 1111 1010 0000 000
```

10. (25 points) Consider the following ARM program. Trace the values of the registers shown as they change during program execution. Also, trace the writes to memory by any `STR` instructions. There may be unused columns or rows in the tables. If you need to add columns or rows, you may do so. `DCD 1` reserves one word of storage and sets it equal to 1. `SPACE 3` reserves 3 bytes of memory but does not give those bytes a value.

```
        AREA    PROB_10, CODE, READONLY
        ENTRY
        ADR     r10, x                  ; (0)
        MOV     r9, #0                  ; (4)
        ADR     r5, p                   ; (8)
        LDR     r0, i                   ; (12)
        LDR     r1, size                ; (16)
loop    CMP     r0, r1                  ; (20)
        BGE     done                    ; (24)
        SUB     r2, r1, r0              ; (28)
        SUB     r2, r2, #1              ; (32)
        LDR     r3, [r10, r0, LSL #2]   ; (36)
        LDR     r4, [r10, r2, LSL #2]   ; (40)
        CMP     r3, r4                  ; (44)
        STRNE   r9, p                ; (48)
        ADD     r0, r0, #1              ; (52)
        B       loop                    ; (56)
done    B       done                    ; (60)
size    DCD     5                       ; (64)
x       DCD     13000, 298, -4730, 698, 698 ; x has addresses (68-87)
p       DCD     1                       ; (88)
i       DCD     0                       ; (92)
        END
```

r0	**0**	**1**	**2**	**3**	**4**	5			
r1	**5**								
r2	**5**	**4**	**3**	**2**	**1**	0			
r3	**13000**	**298**	**-4730**	**698**	**698**				
r4	**698**	**698**	**-4730**	**298**	**13000**				
r5	**88**								
r9	**0**								
r10	**68**								

Results of any `STR` instructions.

Memory Address	Contents
88	**0**
88	**0**
88	**0**
88	**0**

11. (20 points) Complete the ARM assembly language program below so that it implements the following C++ statements.

```
const int size = 10;
int x[size] = {8, 2, 9, 6, 7, 0, 1, 3, 5, 4};
int y[size] = {399, -87, 12, 0, 42, -367, 57, 92, -1000, 25};
for i = 0; i < 10; i++)
  z[i] = y[x[i]] + 20;
```

```
        AREA    PROB_11, CODE, READONLY
        ENTRY
        ADR     r10, x
        LDR     r0, i
        LDR     r1, size
        ADR     r11, y
        ADR     r12, z
loop    CMP     r0, r1
        BGE     done
        LDR     r2, [r10, r0, LSL #2]
        LDR     r2, [r11, r2, LSL #2]
        ADD     r2, r2, #20
        STR     r2, [r12, r0, LSL #2]
        ADD     r0, r0, #1
        B       loop
done    B       done
x       DCD     8, 2, 9, 6, 7, 0, 1, 3, 5, 4
y       DCD     399, -87, 12, 0, 42, -367, 57, 92, -1000, 25
size    DCD     10
z       SPACE   40
i       DCD     0
        END
```