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

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

1. (1 point) An ARM processor has _**16**_ registers.
2. (1 point) Each register holds _**32**_ bits.
3. (1 point) In the ARM statement, `ADD  r0, r1, r2,` _**r0**_ is the destination register.
4. (1 point) _**RTN**_ is a notation to define operations.
5. (1 point) The _**program counter or PC**_ is a register that holds the address of the instruction currently being executed.

6. (10 points) Convert decimal +508 and -193 to binary, using signed-2's complement representation and enough digits to accommodate the numbers.

```
        0    1
16      1   15                          0    12
16     31   12                   16     12     1
16    508                        16    193
+508 = 0x1FC = 0001 1111 1100 in 12 bits, the minimum number necessary
to represent +508 is 10 bits
+193 = 0xC1 = 0 1100 0001
-193 = 1 0011 1111
```

7. (3 points) What is the decimal equivalent of 010100_4 (assume positional notation and unsigned integer formats)?

$010100_4 = 0 \times 4^5 + 1 \times 4^4 + 0 \times 4^3 + 1 \times 4^2 + 0 \times 4^1 + 0 \times 4^0 = 0 + 256 + 0 + 16 + 0 + 0 = 272$

8. (12 points) If `r1 = 0x000F 0FFF` and `r2` = 4, 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
__________________________________________________________
                  0000 0111 1001 0111 0000 1111 0111 1111 (0x0797 0F7F)

(b)   ADD r0, r1, r1, ROR #3
 r1               0000 0000 0000 1111 0000 1111 1111 1111
+r1 ROR #3        1110 0000 0000 0001 1110 0001 1111 1111
__________________________________________________________
                  1110 0000 0001 0000 1111 0001 1111 1110 (0xE010 F1FE)

(c)   ADD r0, r1, r1, LSR r2
 r1               0000 0000 0000 1111 0000 1111 1111 1111
+r1 LSR r2        0000 0000 0000 0000 1111 0000 1111 1111
__________________________________________________________
                  0000 0000 0001 0000 0000 0000 1111 1110 (0x0010 00FE)
```

9. (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) 110000 - 000001

```
110000    C
 110000
 111111
 101111   S

C = 1 (C from MSB)
Z = 0 (101111 != 000000)
V = 0 (C from MSB ⊕ C into MSB)
N = 1 (Result MSB)
```

(b) 011111 + 011111

```
011111     C
 011111
 011111    S
 111110

C = 0 (C from MSB)
Z = 0 (101111 != 000000)
V = 1 (C from MSB ⊕ C into MSB)
N = 1 (Result MSB)
```

10. (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 `0xFF00 1110` and that `r0` contains `0x0F9F 5400`. Use these initial values for each instruction individually.

`(a) LDR   r1, [r2]`

Register Transfer	_r1 ← M[r2]_
r2	_0xFF00 1110_
Effective Address	_0xFF00 1110_

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

Register Transfer	_M[r2 + 13] ← r1_
r2	_0xFF00 1110_
Effective Address	_0xFF00 111D_

`(c) LDR   r1, [r2, #0x2C]!`

Register Transfer	_r2 ← r2 + 44, r1 ← M[r2]_
r2	_0xFF00 113C_
Effective Address	_0xFF00 113C_

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

Register Transfer	_M[r2] ← r1, r2 ← r2 -4_
r2	_0xFF00 110C_
Effective Address	_0xFF00 1110_

`(d)   LDR  r1, [r2, r0, ASR #3]`

Register Transfer	_r1 ← M[r2 + r0 >> 3]_
r2	_0xFF00 1110_
Effective Address	_0xFF00 1110 + 0x01F3 EA80 = 0x00F3 FB90_

11. (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_11, CODE, READONLY
        ENTRY
        ADR    r10, x                  ; 0
        ADR    r11, y                  ; 4
        LDR    r0, size                ; 8
        MOV    r1, #0                  ; 12
        MOV    r2, #5                  ; 16
Loop    CMP    r1, r0                  ; 20
        BGE    done                    ; 24
        SUB    r3, r0, r1              ; 28
        STR    r3, [r10, r1, LSL #2]   ; 32
        SUB    r3, r2, r1              ; 36
        STR    r1, [r11, r3, LSL #2]   ; 40
        ADD    r1, r1, #1              ; 44
        B      loop                    ; 48
done    B      done                    ; 52
size    DCD    6                       ; 56
x       SPACE  24                      ; 60
y       SPACE  24                      ; 84
        END
```

r0	6													
r1	0	1	2	3	4	5	6							
r2	5													
r3	6	5	4	3	2	1	0							
r10	60													
r11	84													

Results of any STR instructions.

Memory Address	Contents
60	6
104	0
64	5
100	1
68	4
96	2
72	3
92	3
76	2
88	4
80	1
84	5

12. (20 points) Complete the ARM assembly language program below so that it implements the following C++ statements.

```
const int size = 10;
int x[size];
int flip = 1;
for i = 0; i < size; i++)
  if (x[i] != x[size - i - 1])
    flip = 0;
```

```
        AREA   PROB_12, CODE, READWRITE
        ENTRY
        ADR    r10, x                  ; put the address of x in r10
        LDR    r0, i                   ; initialize i to 0
        LDR    r1, size                ; initialize r1 to size
        MOV    r11, #1                 ; put the constant 1 in r11
        STR    r11, flip               ; in initialize
        MOV   r9, #0                   ; r9 = 0 for storing in flip, if needed
loop    CMP   r0, r1                   ; compare i and size
        BGE   done                     ; end loop if i >= size
        SUB   r2, r1, r0               ; r2 gets size - i
        SUB   r2, r2, #1               ; r2 gets size - i - 1
        LDR   r3, [r10, r0, LSL #2]    ; r3 gets x[i]
        LDR   r4, [r10, r2, LSL #2]    ; r4 gets x[size - i - 1]
        CMP   r3, r4                   ; compare x[i] and x[size - i - 1]
        STRNE r9, flip                 ; if they are not equal, make flip 0
        ADD   r0, r0, #1               ; i++
        B     loop                     ; next iteration
done    B     done                     ;
x       DCD    13000, 298, -4729, 698, 731, -57 698, -4729, 298, 13000
size    DCD    10
flip    SPACE   4
i       DCD     0
        END
```