The University of Alabama in Huntsville
Electrical and Computer Engineering Department
CPE 221 01
Final Exam
Fall 2018

This test is closed book, closed notes. You may use a calculator. You should have the ARM reference packet. You must show your work to receive full credit.

1. (1 point) _**Volatile**_ memories retain stored data only as long as they receive electric power.
2. (1 point) _**DRAM**_ employs a semiconductor technology that stores data as an electrostatic charge in a capacitor.
3. (1 point) Address _**decoding**_ deals with the way in which address components are mapped onto a processor's physical address space.
4. (1 point) _**Abstract**_ RTL is implementation independent.
5. (1 point) (True or False) _**False**_ A two-way set associative cache has two sets.
6. (4 points) In an ARM computer, r2 contains a value of -925 in decimal. What is the binary value of r1 after this instruction is executed?

```
LSR r1, r2, #7
```
925 = 3 x 256 + 9 x 16 + 13 = 0000 0000 0000 0000 0000 0011 1001 1101
-925 = 1111 1111 1111 1111 1111 1100 0110 0011
r1 = 0000 0001 1111 1111 1111 1111 1111 1000 = 0x01FF FFF8

7. (4 points) In an ARM computer, r2 contains a value of 497 in decimal. What is the binary value of r2 after this instruction is executed?

```
MOVT   r2, #497
```
497 = 1 x 256 + 15 x 16 + 1 = 0000 0000 0000 0000 0000 0001 1111 0001
r2 = 0000 0001 1111 0001 0000 0001 1111 0001 = 0x01F1 01F1

8. (2 points) In an ARM computer, r2 contains a value of -925 in decimal while r3 contains a value of 497 in decimal. What is the binary value of r1 after this instruction is executed?

```
ORN   r1, r2, r3
r3           = 0000 0000 0000 0000 0000 0001 1111 0001
NOT r3       = 1111 1111 1111 1111 1111 1110 0000 1110
r2           = 1111 1111 1111 1111 1111 1100 0110 0011
r2 OR NOT r3 = 1111 1111 1111 1111 1111 1110 0110 1111 = 0xFFFF FE6F
```

9. (13 points) (a) (8 points) What are the values of the following registers when the program executes "`48 B loop`" for the sixth time? Answer in decimal.

r2: **37** r4 **7** r10: **88** r1: **10**

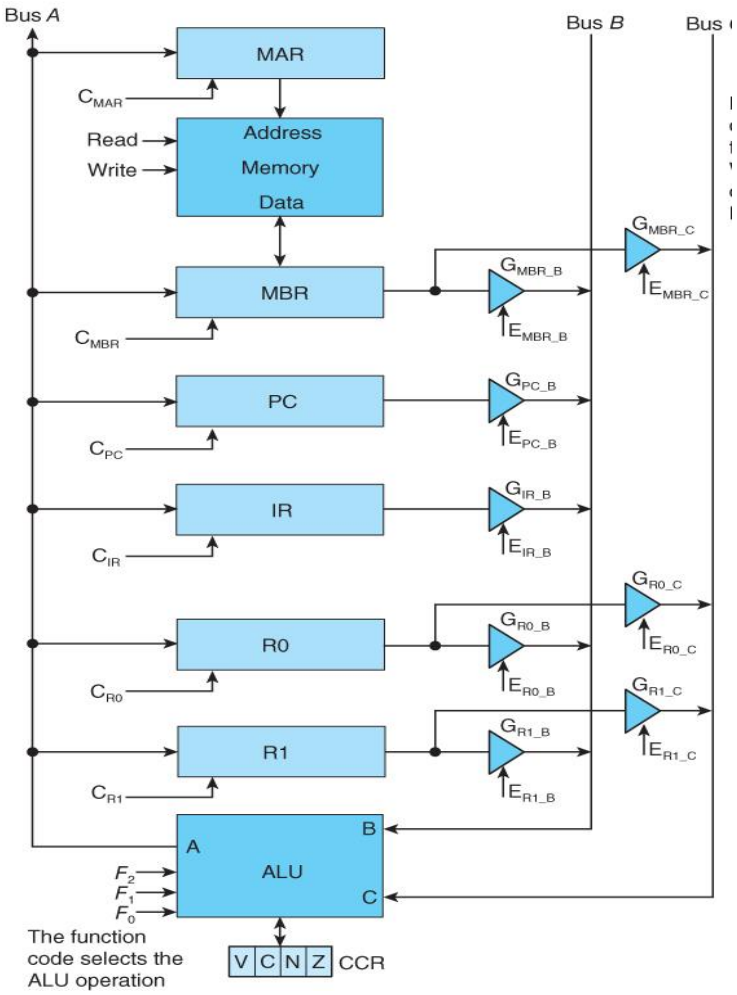
(b) (5 points) What value is written by the `48 STR r2, max` instruction? Answer in decimal.

`48 STR r2, max` **37**

```
            AREA PROB_9, CODE, READONLY
            ENTRY

0           ADR   r10, array
4           LDR   r1, size
8          LDR   r2, [r10]
12          MOV   r4, #1
16   loop   SUBS  r5, r4, r1
20          BPL   done
24          LDR   r7, [r10, #4]!
28          SUBS  r8, r7, r2
32          BMI   next
36          MOV   r2, r7
40   next   ADD   r4, r4, #1
44          B     loop
48   done   STR   r2, max
52   stop   B     stop
56   size   DCD   10
60   max    SPACE 4
64   array  DCD   5, 3, -1, 2, 4, 37, -100, 13, -5, 0
            END
```

10. (15 points) For the architecture shown, write the concrete RTL and the sequence of signals and control actions necessary to execute the instruction `STRM P, R1, R0`, that stores R1 – R0 in the memory location P. Assume that the address P is in the instruction register, IR.
Abstract RTL: M[P] ← R1 – R0



F_1	F_0	Operation
0	0	A = B′
0	1	A = B
1	0	A = B + C
1	1	A = B + 1

Cycle	Concrete RTL	Signals
1	**MAR ← IR**	**E_{IR_B}, F = 01, C_{MAR}**
2	**MBR ← R0′**	**E_{R0_B}, F = 00, C_{MBR}**
3	**MBR ← MBR + 1**	**E_{MBR_B}, F = 11, C_{MBR}**
4	**MBR ← MBR + R1**	**E_{MBR_B}, E_{R1_C}, F = 10, C_{MBR}**
5	**M[MAR] ← MBR**	**Write**

11. (10 points) A certain memory system has a 32 GB main memory and a 128 MB cache. Blocks are 4 words and each word is 32 bits. Show the fields in a memory address if the cache is 2-way set associative. This memory system is byte addressable.

128 MB x 1 word/4 bytes x 1 block/4 words x 1 set/2 blocks = 4M sets, index = 22 bits
Byte offset = 2 bits, 4 bytes per word
Block offset = 2 bits, 4 words per block
Address is 35 bits ($\log_2$ 32 GB)
Tag = 35 – (index + block offset + byte offset) = 35 – (22 + 2 + 2) = 35 – 26 = 9 bits

12. (6 points) You are tasked with building a memory with 35 bits of address so that there are a total of 2^{35} data words. Each data word consists of 64 bits. The only parts you have available to you are static RAM chips that have 18 address bits, so that there are a total of 2^{18} entries. Each chip outputs 8 bits of data. (a) (2 points) How many rows are required? (b) (2 points) How many columns are required? (c) (2 points) How many chips in all?
(a) 2^{35} total address space/2^{18} address space of each chip = 2^{35-18} = 2^{17} rows
(b) 64 total data bits/8 data bits per chip = 8 columns
(c) 2^{17} rows * 8 chips per row = 2^{20} total chips

13. (6 points) A RISC processor executes the following code. There are data dependencies. A source operand cannot be used until it has been written.

```
LDR   r2, [r4]
MOV   r3, r2
STR   r6, [r2]
```

Assuming a five-stage pipeline (fetch (IF), operand fetch (OF), execute (E), memory access (M), and register write (W)), how many extra cycles are required to ensure that the correct value of r2 is available for the `MOV` instruction? **3 extra cycles are required**

	1	2	3	4	5	6	7	8	9	10	11
`LDR r2, [r4]`	**IF**	**OF**	**E**	**M**	**W**						
`MOV r3, r2`		**IF**	**Extra**	**Extra**	**Extra**	**OF**	**E**	**M**	**W**		
`STR r6, [r2]`						**IF**	**OF**	**E**	**M**	**W**	

14. (20 points) Complete the ARM assembly language program below so that it implements the following C++ statements.

```
;
;       This program adds two arrays, element by element into a third
;       array. It also writes a 0 into an array called sign if the sum
;       sum is negative and a 1 into the array called sign if the sum
;       is positive.
;
;       const int size = 10;
;       int x[size] = {100, 3, -1, 2, 4, 4, 2, -1, 3, 100};
;       int y[size] = {-53, 247, 95, -7, 481, 91, -33, 1500, 29, -83};
;       int z[size];
;       int sign[size];
;       int i;
;       for (i = 0; i < size; i++)
;       {
;         z[i] = x[i] + y[i];
;         if (z[i] < 0)
;           sign[i] = 0;
;         else
;           sign[i] = 1;
;       }


        AREA    PROB_11, CODE, READEXECUTE
        ENTRY
        ADR     r0, x
        ADR     r1, y
        ADR     r2, z
        LDR     r3, size
        LDR     r4, i
        ADR     r5, sign
loop    CMP     r4, r3
        BPL     done
        LDR     r6, [r0, r4 LSL #2]
        LDR     r7, [r1, r4 LSL #2]
        ADDS    r8, r6, r7
        STR     r8, [r2, r4 LSL #2]
        MOVPL   r10, #1
        MOVMI   r10, #0
        STR     r10, [r5, r4 LSL #2]
        ADD     r4, r4, #1
        B       loop
done    B       done
x       DCD     100, 3, -1, 2, 4, 4, 2, -1, 3, 100
y       DCD     -53, 247, 95, -7, 481, 91, -33, -1500, 29, -83
z       SPACE    40
sign    SPACE    40
i       DCD      0
size    DCD      10
        END
```

15. (15 points) Consider the following ARM program. Trace the stack activity, including all changes to the stack pointer and to the contents of the stack. Clearly indicate the value of the sp.

```
0          MOV     sp, #0x00000000
4          B       main
8    swap  SUB     sp, sp, #4
12         LDR     r1, [sp, #8]
16         LDR     r2, [r1]
20         STR     r2, [sp]
24         LDR     r0, [sp, #4]
28         LDR     r3, [r0]
32         STR     r3, [r1]
36         LDR     r3, [sp]
40         STR     r3, [r0]
44         ADD     sp, sp, #4
48         MOV     pc, lr
52   main  SUB     sp, sp, #8
56         ADR     r6, x
60         STR     r6, [sp, #4]
64         ADR     r6, y
68         STR     r6, [sp]
72         BL      swap
76         ADD     sp, sp, #8
80   stop  B       stop
84   x     DCD     2
88   y     DCD     3
```

Address	Value
FFFF FFF0	
FFFF FFF4	
FFFF FFF8	
FFFF FFFC	

Instruction:
0 MOV sp, #0, sp = 0

Address	Value
FFFF FFF0	
FFFF FFF4	
FFFF FFF8	
FFFF FFFC	

Instruction:
52 SUB sp, sp, #8

Address	Value
FFFF FFF0	
FFFF FFF4	
FFFF FFF8	
FFFF FFFC	**84**

Instruction:
__60 STR r6, [sp, #4]__

Address	Value
FFFF FFF0	
FFFF FFF4	
FFFF FFF8	**88**
FFFF FFFC	**84**

Instruction:
68 STR r6, [sp]

Address	Value
FFFF FFF0	
FFFF FFF4	
FFFF FFF8	**88**
FFFF FFFC	**84**

Instruction:
__8 SUB sp, sp, #4__

Address	Value
FFFF FFF0	
FFFF FFF4	**2**
FFFF FFF8	**88**
FFFF FFFC	**84**

Instruction:
__20 STR r2, [sp]_

Address	Value
FFFF FFF0	
FFFF FFF4	**2**
FFFF FFF8	**88**
FFFF FFFC	**84**

Instruction:
__44 ADD sp, sp, #4_

Address	Value
FFFF FFF0	
FFFF FFF4	**2**
FFFF FFF8	**88**
FFFF FFFC	**84**

Instruction:
_76 ADD sp, sp, #8, sp = 0__

Address	Value
FFFF FFF0	
FFFF FFF4	
FFFF FFF8	
FFFF FFFC	

Instruction:
_____

Note: Blue shaded address indicates the value of the stack pointer.