Homework 3: ARM Basics
CPE221

Instructor: Rahul Bhadani

Due: Feb 14, 2025, 11:59 PM
100 points

You are allowed to use a generative model-based AI tool for your assignment. However, you must submit an accompanying reflection report detailing how you used the AI tool, the specific query you made, and how it improved your understanding of the subject. You are also required to submit screenshots of your conversation with any large language model (LLM) or equivalent conversational AI, clearly showing the prompts and your login avatar. Some conversational AIs provide a way to share a conversation link, and such a link is desirable for authenticity. Failure to do so may result in actions taken in compliance with the plagiarism policy.

Additionally, you must include your thoughts on how you would approach the assignment if such a tool were not available. Failure to provide a reflection report for every assignment where an AI tool is used may result in a penalty, and subsequent actions will be taken in line with the plagiarism policy.

Submission instruction:

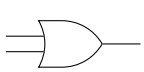
Upload a .pdf on Canvas with the format {firstname_lastname}_CPE221_hw03.pdf. For example, if your name is Sam Wells, your file name should be sam_wells_CPE221_hw03.pdf. If there is a programming assignment, then you should include your source code along with your PDF files in a zip file {firstname_lastname}_CPE221_hw03.zip. Your submission must contain your name, and UAH Charger ID or the UAH email address. Please number your pages as well.

---

1 IEEE 754 Floating-point Standard(10 points)

Convert the following decimal numbers to the binary representation using the IEEE 754 single precision format. Show sign, exponent, and mantissa separately. For example, -19 has sign of -1, exponent 2^4, and mantissa 1.1875.

1. -1609.5
2. -938.8125



3. 130.59375
4. -37.3125
5. 2.375

Answer

Single Precision:

1. 1100 0100 1100 1001 0011 0000 0000 0000
   Sign:-1, Exponent: 2^{10}, Mantissa: 1.57177734375.
2. 1100 0100 0110 1010 1011 0100 0000 0000
   Sign:-1, Exponent: 2^{9}, Mantissa: 1.8336181640625.
3. 0100 0011 0000 0010 1001 1000 0000 0000
   Sign:$+1$, Exponent: 2^{7} , Mantissa: 1.020263671875.
4. 1100 0010 0001 0101 0100 0000 0000 0000
   Sign:-1, Exponent: 2^{5}, Mantissa: 1.166015625.
5. 0100 0000 0001 1000 0000 0000 0000 0000
   Sign: $+1$, Exponent:2^{1} , Mantissa: 1.1875.

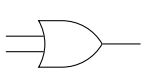
2 Flag it! (10 points)

For each of the following 8-bit operations, calculate the values of the C, Z, V, and N flags.

1. 1010 1101 + 1010 1101
2. 1111 1111 + 0100 0001
3. 0110 0100 - 1110 1101
4. 0010 1111 + 0110 1111
5. 0100 1000 – 1110 1001

Answer

1. C = 1, Z = 0, V = 1 $\oplus$ 0 = 1, N = 0
2. C = 1, Z = 0, V = 1 $\oplus$ 1 = 0, N = 0



3. C = 0, Z = 0, V = 0 $\oplus$ 0 = 0, N = 0
4. C = 0, Z = 0, V = 0 $\oplus$ 1 = 1, N = 1
5. C = 0, Z = 0, V = 0 $\oplus$ 0 = 0, N = 0

3 On the way to be a hacker! (10 points)

If R1 = 0xA536 2C89 what is the value of R0 after each of the following instructions has been executed (assume that each instruction uses the same data)?

1. ADD R0, R1, R1, LSL #4
2. ADD R0, R1, R1, ASR #8
3. ADD R0, R1, R1, ROR #9
4. If R2 = 0xB903 8471 and R3 = -160, Write down effective address is generated by the instruction

   LDR R0, [R2, R3, LSL #1].

   Express your answer in hexadecimal.

Answer

1. ADD R0, R1, R1, LSL #4

   R1 = 1010 0101 0011 0110 0010 1100 1000 1001
   R1 LSL #4 = 0101 0011 0110 0010 1100 1000 1001 0000

   ADD = 1111 1000 1001 1000 1111 0101 0001 1001

   Final answer: 0xF898 F519

2. ADD R0, R1, R1, ASR #8

   R1 = 1010 0101 0011 0110 0010 1100 1000 1001
   R1 ASR #8 = 1111 1111 1010 0101 0011 0110 0010 1100

   ADD = 1010 0100 1101 1011 0110 0010 1011 0101

   Final answer: 0xA4DB 62B5

3. ADD R0, R1, R1, ROR #9

R1	= 1010 0101 0011 0110 0010 1100 1000 1001
R1 ROR #7	= 0100 0100 1101 0010 1001 1011 0001 0110
ADD	= 1110 1010 0000 1000 1100 0111 1001 1111

Final answer: 0xEA08 C79F

4. LDR R0, [R2, R3, LSL #1]

   160 = 0x0000 00A0

   -160 = 0x FFFF FF60

R2	= 1011 1001 0000 0011 1000 0100 0111 0001
R3 LSL #1	= 1111 1111 1111 1111 1111 1110 1100 0000
ADD	= 1011 1001 0000 0011 1000 0011 0011 0001

   Final answer: 0xB903 8331

4 Decipher (10 points)

Explain the effect of each of the following six instructions. Use register transfer notation.

1. LDR R1, [R4]
2. LDR R1, [R4, #12]
3. STR R1, [R2, #16]
4. STR R1, [R2, R1, ASR #4]
5. LDR R1, [R2, R1, LSL #9]

Answer

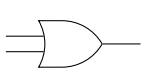
1. R1 ← M[R4]

   It moves the value stored in the memory address saved in the register R4 to the register R1.

2. R1 ← M[R4 + 12]

   It moves the value stored in the memory address (saved in the register R4 plus 12, i.e. offset by 12 places) to the register R1.

3. M[R2 + 16] ← R1



Saves the value of the register R1 in the memory location pointed by the value of register R2 plus 16 places of offset

4. M[R2 + (R1 $>>$ 4)] $\leftarrow$ R1 **($>>$ denotes a shift right)**

   First, the memory address value stored in the register R1 is shifted to the right by 4 places (in bits), then added to the value of the register R2, which acts as a new target memory location where the value from the register R1 is saved.

   Note that we are using ASR so when the right shift happens, all MSBs are filled with the same original MSB digit.

5. R1 $\leftarrow$ M[R2 + (R1 $<<$ 9)] **($<<$ denotes a shift left)**

   The memory location value saved in register R1 is shifted to the left by 9 places (in bits) then the value is added to the value of the register R1, then the sum is used as a new memory address from where the value is fetched and loaded to the register R1.

5 No Shortcut (10 points)

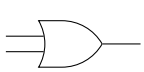
Without using the ARM's multiplication instruction, write one or more instructions (using ADD, SUB, and shifting) to multiply the contents of R1 by 145 and put the result in R0

Answer

```
1 MOV R0, #0
2 ADD R0, R0, R1, LSL #7 @ R0 = 128 * R1
3 ADD R0, R0, R1, LSL #4 @ R0 = 144 * R1
4 ADD R0, R0, R1 @ R0 = 145 * R1
```

6 Closer to being a Robot! (20 points)

Write ARM assembly code to implement the following C statements, assuming all variables are 32-bit integers and a declaration of $x[10]$, $y[10]$ and size $= 10$. Make sure to add comments in your assembly code documenting what your assembly code means and what they are trying to do. Comments in assembly code start '@'. You can assume about the pointer to the first element of a register storing x, y, and other constants.



```
int x[10] = {8, 2, 9, 6, 7, 0, 1, 3, 5, 4};
for(int i = 0; i < 10; i++)
{
    y[i] = x[i] + i;
}
```

Answer

```
.global _start
_start:

        LDR R0, =x @load the address of x
        LDR R1, =y @load the address of y
        LDR R4, =size @ load the address of size
        LDR R4, [R4] @ load the size = 10
        MOV R5, #0 @counter
loop:
        CMP R5, R4
         BGE done @ if i>= size, done
        LSL R6, R5, #2 @calculating offset for the array e.g. 4^i
        LDR R2, [R0, R6] @load the value from x[i]
        ADD R7, R2, R5 @  x[i] + i
        STR R7, [R1, R6] @ y[i] = x[i] + i
        ADD R5, R5, #1 @ i = i + 1
         B loop @
done: B done @


size:  .word 10
x:  .word 8, 2, 9, 6, 7, 0, 1, 3, 5, 4
y:  .space 40
i:  .word 0

.end
```


7 Am I a Hacker Now! (20 points)

Write an equivalent C code for the following: (Note: MVN is the bitwise-NOT operation in ARM assembly.)

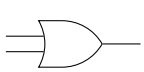
1.
```
MOV R6, R4
LSL R6, R6, #2
SUB R6, R4, R6
MOV R8, R6
```

2.
```
MOV R3, R4
MUL R5, R4, #4
ASR R3, R3, #3
ADD R3, R3, R5
MOV R8, R3
```

3.
```
MOV R10, R4
LSL R2, R4, #2
LSL R2, R2, R4
MOV R8, R2
```

4.
```
MOV R9, R4
MVN R9, R9
AND R9, R9, R4
MOV R8, R9
```

5.
```
MOV R12, R4
AND R12, R12, #15
ASR R1, R4, #2
ADD R12, R12, R1
MOV R8, R12
```

Answer

Assuming these registers contain the specified variables: R1: t R2: u R3: w R4: x R5: v R6: y R8: z R9: p R10: q R12: r

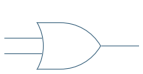
1.
```
y = x;
y = y << 2;
y = x - y;
z = y;
```

2.
```
w = x;
v = x*4;
w = w >> 3; // C does the arithmetic shift
w = w + v;
z = w;
```

3.
```
q = x;
u = x << 2;
u = u << x;
z = u;
```

4.
```
p = x;
p = ~p;
p = p & x;
z = p;
```

5.
```
r = x;
r = r & 15;
t = x >> 2;
r = r + t;
z = r;
```

8 Being a Polyglot in Computing. (10 points)

Interpret the following Assembly code and write the value of each register at every line.

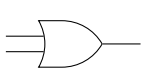
```
1     MOV R5, #0
2     MOV R6, #1
3     MOV R1, #2
4 LOOP:
5     CMP R1, #15
6     BGE EXIT_LOOP
7
8     ADD R7, R5, R6
9     MOV R5, R6
10    MOV R6, R7
11    ADD R1, R1, #1
12    B LOOP
13 EXIT_LOOP
14
```

Answer

Assuming R5: x, R6: y, R1: z, R7: p, an equivalent C code is below (points not deducted if C code is not provided):

```
x = 0;
y = 1;
z = 2;
while (z < 15)
{
   p = x + y;
   x = y;
   y = p;
   z = z  + 1;
}
```

The value of registers at lines 1 - 3 are 0, 1, 2. The loop will iterate for a total of 13 times. Line 6 will not change the value of the register.
Iteration: 1



Line 8: R1 = 2, R5 = 0, R6 = 1, R7 = 1
Line 9: R1 = 2, R5 = 1, R6 = 1, R7 = 1
Line 10: R1 = 2, R5 = 1, R6 = 1, R7 = 1
Line 11: R1 = 3, R5 = 1, R6 = 1, R7 = 1

Iteration: 2
Line 8: R1 = 3, R5 = 1, R6 = 1, R7 = 2
Line 9: R1 = 3, R5 = 1, R6 = 1, R7 = 2
Line 10: R1 = 3, R5 = 1, R6 = 2, R7 = 2
Line 11: R1 = 4, R5 = 1, R6 = 2, R7 = 2

Iteration: 3
Line 8: R1 = 4, R5 = 1, R6 = 2, R7 = 3
Line 9: R1 = 4, R5 = 2, R6 = 2, R7 = 3
Line 10: R1 = 4, R5 = 2, R6 = 3, R7 = 3
Line 11: R1 = 5, R5 = 2, R6 = 3, R7 = 3

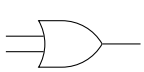
Iteration: 4
Line 8: R1 = 5, R5 = 2, R6 = 3, R7 = 5
Line 9: R1 = 5, R5 = 3, R6 = 3, R7 = 5
Line 10: R1 = 5, R5 = 3, R6 = 5, R7 = 5
Line 11: R1 = 6, R5 = 3, R6 = 5, R7 = 5

Iteration: 5
Line 8: R1 = 6, R5 = 3, R6 = 5, R7 = 8
Line 9: R1 = 6, R5 = 5, R6 = 5, R7 = 8
Line 10: R1 = 6, R5 = 5, R6 = 8, R7 = 8
Line 11: R1 = 7, R5 = 5, R6 = 8, R7 = 8

Iteration: 6
Line 8: R1 = 7, R5 = 5, R6 = 8, R7 = 13
Line 9: R1 = 7, R5 = 8, R6 = 8, R7 = 13
Line 10: R1 = 7, R5 = 8, R6 = 13, R7 = 13
Line 11: R1 = 8, R5 = 8, R6 = 13, R7 = 13

Iteration: 7
Line 8: R1 = 8, R5 = 8, R6 = 13, R7 = 21
Line 9: R1 = 8, R5 = 13, R6 = 13, R7 = 21
Line 10: R1 = 8, R5 = 13, R6 = 21, R7 = 21
Line 11: R1 = 9, R5 = 13, R6 = 21, R7 = 21

Iteration: 8
Line 8: R1 = 9, R5 = 13, R6 = 21, R7 = 34
Line 9: R1 = 9, R5 = 21, R6 = 21, R7 = 34



Line 10: R1 = 9, R5 = 21, R6 = 34, R7 = 34
Line 11: R1 = 10, R5 = 21, R6 = 34, R7 = 34

Iteration : 9
Line 8: R1 = 10, R5 = 21, R6 = 34, R7 = 55
Line 9: R1 = 10, R5 = 34, R6 = 34, R7 = 55
Line 10: R1 = 10, R5 = 34, R6 = 55, R7 = 55
Line 11: R1 = 11, R5 = 34, R6 = 55, R7 = 55

Iteration: 10
Line 8: R1 = 11, R5 = 34, R6 = 55, R7 = 89
Line 9: R1 = 11, R5 = 55, R6 = 55, R7 = 89
Line 10: R1 = 11, R5 = 55, R6 = 89, R7 = 89
Line 11: R1 = 12, R5 = 55, R6 = 89, R7 = 89

Iteration: 11
Line 8: R1 = 12, R5 = 55, R6 = 89, R7 = 144
Line 9: R1 = 12, R5 = 89, R6 = 89, R7 = 144
Line 10: R1 = 12, R5 = 89, R6 = 144, R7 = 144
Line 11: R1 = 13, R5 = 89, R6 = 144, R7 = 144

Iteration: 12
Line 8: R1 = 13, R5 = 89, R6 = 144, R7 = 233
Line 9: R1 = 13, R5 = 144, R6 = 144, R7 = 233
Line 10: R1 = 13, R5 = 144, R6 = 233, R7 = 233
Line 11: R1 = 14, R5 = 144, R6 = 233, R7 = 233

Iteration: 13
Line 8: R1 = 14, R5 = 144, R6 = 233, R7 = 377
Line 9: R1 = 14, R5 = 233, R6 = 233, R7 = 377
Line 10: R1 = 14, R5 = 233, R6 = 377, R7 = 377
Line 11: R1 = 15, R5 = 233, R6 = 377, R7 = 377
If you carefully notice, then R7 yields the Fibonacci sequence.