

CSN 6114 Computer Architecture & Organization**Assignment 2 (10%): ARM Programming****Instructions**

This is a group assignment consist of **FOUR** students from the same tutorial group. You may opt to do it alone however no extra marks will be given.

Write **ARM instructions** to do the following:

The list of instruction supported by ARM emulator is stated in

https://salmanarif.bitbucket.io/visual/supported_instructions.html

Your question will be based the following criteria

- a) $(\text{FirstStudentID} + \text{SecondStudentID} + \text{ThirdStudentID} + \text{FourthStudentID}) \% 30 = \text{Question No listed in your tutorial section.}$

Student must do the question assigned to them based on the above formula. Otherwise, your assignment will not be graded. Please choose your partner wisely as this may affect the question you need to do. If you result of the modulus operation is zero, then you should do Question No#30.

Initialization data.

Add this initialization Code at the beginning of your program

;data initialization. 10 random no (4 byte each stored starting from address 0x2000h). These numbers will be randomly changed to test the correctness of your program

; Put your name and ID here. Student1 name, student1 ID.

; StudentN name, StudentN ID....

```

        mov        r0,#0x11000000
        mov        r1,#0x00110000
        mov        r2,#0x00001100
        mov        r3,#0x00000011
        add        r0, r0, r1
        add        r0, r0, r2
        add        r0, r0, r3
        mov        r8,#0x2000
        str        r0, [r8]

;
        move       r0,#0x22223333
        mov        r0,#0x22000000
        mov        r1,#0x00220000
        mov        r2,#0x00003300
        mov        r3,#0x00000033
        add        r0, r0, r1
        add        r0, r0, r2
        add        r0, r0, r3
        str        r0, [r8,#4]

;
        move       r0,#0x31111111

```

```

mov        r0,#0x31000000
mov        r1,#0x00110000
mov        r2,#0x00001100
mov        r3,#0x00000011
add        r0, r0, r1
add        r0, r0, r2
add        r0, r0, r3
str        r0, [r8,#8]

;          move        r0,#0x42223333
mov        r0,#0x42000000
mov        r1,#0x00220000
mov        r2,#0x00003300
mov        r3,#0x00000033
add        r0, r0, r1
add        r0, r0, r2
add        r0, r0, r3
str        r0, [r8,#12]

;          move        r0,#0x51111111
mov        r0,#0x51000000
mov        r1,#0x00110000
mov        r2,#0x00001100
mov        r3,#0x00000011
add        r0, r0, r1
add        r0, r0, r2
add        r0, r0, r3
str        r0, [r8,#8]
str        r0, [r8,#16]

;          move        r0,#0x62223333
mov        r0,#0x62000000
mov        r1,#0x00220000
mov        r2,#0x00003300
mov        r3,#0x00000033
add        r0, r0, r1
add        r0, r0, r2
add        r0, r0, r3
str        r0, [r8,#12]
str        r0, [r8,#20]

;          move        r0,#0x71111111
mov        r0,#0x71000000
mov        r1,#0x00110000
mov        r2,#0x00001100
mov        r3,#0x00000011
add        r0, r0, r1
add        r0, r0, r2
add        r0, r0, r3
str        r0, [r8,#24]

```

```

;      move      r0,#0x82223333
      mov       r0,#0x82000000
      mov       r1,#0x00220000
      mov       r2,#0x00003300
      mov       r3,#0x00000033
      add       r0, r0, r1
      add       r0, r0, r2
      add       r0, r0, r3
      str       r0, [r8,#28]

;      move      r0,#0x91111111
      mov       r0,#0x91000000
      mov       r1,#0x00110000
      mov       r2,#0x00001100
      mov       r3,#0x00000011
      add       r0, r0, r1
      add       r0, r0, r2
      add       r0, r0, r3
      str       r0, [r8,#32]

;      move      r0,#0xA2223333
      mov       r0,#0xA2000000
      mov       r1,#0x00220000
      mov       r2,#0x00003300
      mov       r3,#0x00000033
      add       r0, r0, r1
      add       r0, r0, r2
      add       r0, r0, r3
      str       r0, [r8,#36]

```

Note: You must change the above test data yourself to suit the question requirement. The lecturer may

- a) change the above test data to test the accuracy of your program.**
- b) Request you to change the test data to specific values written in a piece of paper before running your program**
- c) Request you to show multiple test data that will run correctly and multiple test data which will run incorrectly.**
- d) Request you to show multiple test data that will run partial correct depending on your question. For example, the for date format, the day is correct, month is wrong, year is correct.**

; Add your code below here.

Stored the result of your assignment program in memory location 0x2100 – 2150.

For Q1 to Q5. Student are required to do two part. Assume the data is an unsigned integer of 32 bit for Q1 to Q5.

Q1a. Calculate the sum of all data, store the result in 2100, in (unsigned integer 64bit)

Q1b. Find the largest value, store the result in 2150,

Q2a. Find the smallest value, store the result in 2100.

Q2b. Find the average value, store the result in 2150

Q3a. Find the largest difference, store the result in 2100

Q3b. Find the smallest difference, store the result in 2150

Q4a. Sort the value in ascending order. Store the result in 2100 onwards.

Q4b. Sort the value in descending order. Store the result in 2150 onwards

Q5a. Multiply the data (16-bit value stored in 32 bit word) in 2000 and 2004 and stored the 32bit value in 2100. Tips: (Use shift and Add method).

Q5b. Divide the data in 2000 with 2004, if the value in 2000 is bigger than 2004. Else divide the data in 2004 with 2000. Stored the result in 2150.

Example for division via subtraction:

```

                                mov      r0, #0x340 ; Example r2 = R0/R1
                                mov      r1, #0x34
                                mov      r2, #0
loop1
                                add      r2, r2, #1
                                subs     r0, r0, r1
                                bpl      loop1
                                sub      r2, r2, #1
```

For Q6 to Q13 and Q15. The data is a signed integer of 32 bit. (include +ve and –ve numbers)

Q6. Calculate the sum of all data, store the result in 2100, (64 bit unsigned)

Q7. Find the largest value, store the result in 2100,

Q8. Find the smallest value, store the result in 2100.

Q9. Find the average value, store the result in 2100

Q10. Find the largest difference, store the result in 2100

Q11. Find the smallest difference, store the result in 2100

Q12. Sort the value in ascending order. Store the result in 2100 onwards.

Q13. Sort the value in descending order. Store the result in 2100 onwards

Q14. Multiply the data in (32 bit unsigned integer) 2000 and 2004 and stored it in 2100-2107 as 64 bit unsigned integer.

Q15. Divide the data in 2000 with 2004, if the value in 2000 is bigger than 2004. Else divide the data in 2004 with 2000. Stored the result in 2100.

>>>>>

Q16 to Q20. (assumed the input data is 32 bit unsigned integer).

Q16. Manually enter a 32 bit value in memory location, 0x2200h. Search the value starting from memory location 0x2000 till 0x2000 + 36. Store 0x1111 into 0x2100 if a match is found, Store 0x9999 if a match is not found. Store the number of times a match is found in 0x2104.

Q17. Check the 32 bits value in 0x2000 till 0x2000 + 36 for parity bit. If the data is even parity, store "2". If the data is odd parity, store "1". The result should be stored in 0x2100 till (0x2100 + 36).

*Q18. The 32 bits hexadecimal value stored in 0x2000 is the month. The value must range from 1 to 0xB. The 32 bits hexadecimal value stored in 0x2004 is the day of the month. The value must range from 1 to 31 (decimal) depending on which month. If both input data is not in a correct range, stored "1111" to 0x2100.

If the input data is in correct range, store "2222" to 0x2100. Then, store the number of days since 1st January into 0x2104. For example, 28th Feb should store "2222" in 0x2100 and "59 decimal value" in 0x2104.

Example1: 28:01 0x2000 memory value 28, 0x2004 memory value 2 decimal.

You should store "2222" in 0x2100 and "31+28" or "59 decimal value" in 0x2104.

Example2: 99:38, you should store "1111" in 0x2100. Invalid month and day.

*Q19. The 32 bits hexadecimal value stored in 0x2000 is the minute. The value must range from 0 to "60" decimal. The 32 bits hexadecimal value stored in 0x2004 is the hour of the day. The value must range from 0 to 23 (decimal) depending on which month. If both input data is not in a correct range, stored "1111" to 0x2100.

If the input data is in correct range, store "2222" to 0x2100. Then, store the number of seconds since 00:00 hour.

Example1: 02:11 0x2000 memory value 02, 0x2004 memory value 11 decimal.

You should store "2222" in 0x2100 and "60+60+11" or "131 decimal value" in 0x2104.

Example2: 99:38, you should store "1111" in 0x2100.

*Q20. The 32 bits hexadecimal value stored in 0x2000 is the distance in 1/10th of a millimetre. For example, "10 decimal" = 10 x 1/10 millimetre = 1 millimeter.

"100 decimal" = 1 centimeter

"10000 decimal" = 1 meter.

Note: 254 decimal = 1 inch.

Calculate to the nearest integer the distance in "inch", "feet", "yard" and "miles" and stored it in 0x2100, 0x2104, 0x2108, 0x210C.

For Q21 till Q26. The input is in BCD format. Therefore, each input digit must be from "0" till "9". Input with digit consisting of "A-F" will not be given as test data. If there is "A till F" digit in the test data, change it to digit "1".

Q21. The 32 bits value stored in 0x2000 till 0x2000 + 36 is actually the date in BCD format. For example, DDMMYYYY, for example 0x01021972 is represented as 1st February 1972. If the date format is valid, store "1111" in 0x2100 till (0x2100 + 36). Else store "2222" in 0x2100 till (0x2100 + 36)

Q22. The 32 bits value stored in 0x2000 till 0x2000 + 36 is actually the date in BCD format. For example, MMDDYYYY, for example 0x11221972 is represented as 22nd November 1972. If the date format is valid, store "1111" in 0x2100 till (0x2100 + 36). Else store "2222" in 0x2100 till (0x2100 + 36).

Q23. The 32 bits value stored in 0x2000 till 0x2000 + 36 is actually the date in BCD format. For example, YYYYMMDD, for example 0x19230523 is represented as 23rd May 1923. If the date format is valid, store "1111" in 0x2100 till (0x2100 + 36). Else store "2222" in 0x2100 till (0x2100 + 36).

Q24. The 32 bits value stored in 0x2000 till 0x2000 + 36 is actually the date in BCD format. For example, YYYYDDMM, for example 0x19231503 is represented as 15th March 1923. If the date format is valid, store "1111" in 0x2100 till (0x2100 + 36). Else store "2222" in 0x2100 till (0x2100 + 36).

Q25. The 32 bits value stored in 0x2000 till 0x2000 + 36 is actually the timestamp in BCD format. For example, 00:HH:MM:SEC. For example 0x00233435 is represented as 23 hour 34 min 35 second. If the time format is valid, store "1111" in 0x2100 till (0x2100 + 36). Else store "2222" in 0x2100 till (0x2100 + 36).

Q26. The 32 bits value stored in 0x2000 till 0x2000 + 36 is actually the timestamp in BCD format. For example, 00:SEC:MM:HR. For example 0x00233415 is represented as 15 hour 34 min 23 second. If the time format is valid, store "1111" in 0x2100 till (0x2100 + 36). Else store "2222" in 0x2100 till (0x2100 + 36).

Q27. The 32 bits value stored in 0x2000 till 0x2000 + 36 is in hex value. If input1 = 0x12345678, output = 0x0000 0000. If input = 0xABCDEF00, output is 0xABCDEF00. If input = 0xABCD1234, output is 0xABCD0000. The program will take the input and remove all hex value from 1 to 9. Leaving only the hex value of A,B,C,D,E,F. The result is stored in 0x2100 till (0x2100 + 36)

Q28. The 32 bits value stored in 0x2000 till 0x2000 + 36 is in hex value. If input1 = 0x12345678, output = 0x1030 5070. If input = 0xABCDEF90, output is 0x000 0090. The program will take the input and remove all hex value the following hex value 2,4,6,8,A-F. Leaving only the hex value of 1,3,5,7,9. The result is stored in 0x2100 till (0x2100 + 36)

Q29. The 32 bits value stored in 0x2000 till 0x2000 + 36 is in hex value. If input1 = 0x12345678, output = 0x0204 0608. If input = 0xABCDEF90, output is 0x000 0000. The program will take the input and remove all hex value the following hex value 1,3,5,7,9,A-F. Leaving only the hex value of 2,4,6,8. The result is stored in 0x2100 till (0x2100 + 36)

Q30. The 32 bits value stored in 0x2000 till 0x2000 + 36 is in hex value. If input1 = 0x12345678, output = 0x0204 0608. If input = 0xABCDEF90, output is 0xABCDEF00. The program will take the input and remove all hex value the following hex value 1,3,5,7,9. Leaving only the hex value of 2,4,6,8, A-F The result is stored in 0x2100 till (0x2100 + 36)

For ARM program

You need to submit ARM program file to your tutor according to your tutorial section. Explain your logic / instruction sets by using comment (;).

You need to put student name and ID on top of program.

For example ;Ali bin Abu (1201102449)

;Lee Li Li (1201134567)

File notation: TT???V_Group???

For Report

- i) ARM program
- ii) Snapshot of data and operational result stored in memory address (see Figure 2 for example)

Deadline:

Things to submit to tutor:

1. ARM program file (Part B)
2. Report Part B in PDF

Submission Due date: 30 June 2024 3:00 pm

Zero mark will be given once the submission is closed.