2021 Fall CPSC 240-5 Answers
Midterm #2 Concepts Test
November 10, 2021 1:00pm-3:00pm

Read me

Place your answers in the space following each question.

Place your name in the test in this front page and on the last page.

Near 1:25pm begin to save your test document in either odt, or doc, or docx format.

Before 2:59pm send your document as an attachment to me: holliday@fullerton.edu

If you encounter a question where you feel that you must guess an answer then place the word “Blank” in the space for the answer and you will receive 20% of the credit for that question.

If the answer space is empty then the points for that question are zero.

You may use any word processing tool at your disposal provided it can save files in one of the three accepted formats.

If your computer has no word processor program then try Google Docs, which saves files in every format ever created on this planet.

The total point value of this test is 100 points, which is one-sixth of your course grade.

Every effort has been made to create unambiguous questions. If a question is truly ambiguous send me ordinary email or a chat message to ask for clarification. I will be at computer during the test period probably answering the backlog of email.

This is an open note test.

Proceed to the next page.

1. Who created the world's first computer? [4]

Answer: Charles Babbage

2. What was Richard Stallman's main contribution to the world community of programmers? [4]

Answer: He started the movement that promotes FOSS.
FOSS = Free Open Source Software

3. What is one positive value found in inline assembly programming? [4]

Answer: You can execute blocks of assembly without having access to an assembler like Nasm.

4. Which are the preserved registers? [4]

Answer: rbx, rbp, r12, r13, r14, r15.

Reference: Jorgensen ebook, pages 173-174

5. Suppose centennial is a source file containing a function written in standard C++ syntax. Show how to convert it to an equivalent function written in ATT syntax in a file named millennial.att. [4]

Answer: g++ -m64 -std=c++17 -c -S -o millrnnial.att centennial.cpp

6. Show an assembly instruction that will convert 76 stored in r10 into 76.0 stored in xmm10. [4]

Answer: cvtsi2sd xmm10,r10

Comment: That instruction does a lot of work. The movsd instruction simply copies bits, but cvtsi2sd performs a conversion from twos complement to IEEE.

7. Suppose you are chatting with another assembly programmer during lunch at the big corporation where you both work. He asks you to explain clearly what cwde does. The other programmer wants to discover if that instruction will meet his needs in his current project.

How do you explain cwde? [5]

Answer: The instruction cwde applies only to the register rax.

The lowest 16-bits of rax are a word of data. Using the rules of twos complement integers those 16 bits can be interpreted as a 16-bit signed integer.

The lowest 32 bits of rax are a double word of data.

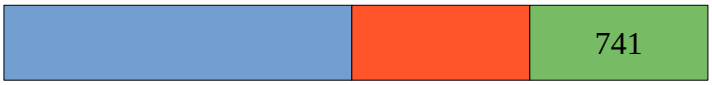
The instruction cwde finds suitable values for bits numbers 31 -16 so that the entire double word is mathematically equal to the original word.

Lets do it by example.

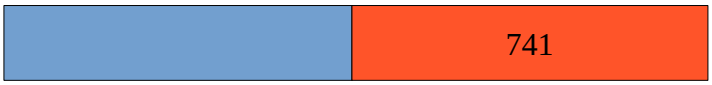
Here is rax = 

The green area is the low word. There are 16 bits in the green area. The rules of twos complement will tell us the decimal value of those 16 bits. Let's say the value is 741.

Consider the 16-bit region immediately higher that the green word: the orange rectangle below.

rax = 

The action of cwde is to find new bits to place into the orange area so the numeric value of the entire low 32 bits shown in orange below has numeric value 741:

rax = 

8. A C++ function is executing and will soon reach the return statement. This is a void function that does not return any value to the caller function. What exactly happens when the return statement is executed? [5]

Solution:

```
mov rsp, rbp
pop rbp
pop rip
```

Comment: The goal upon reaching return is to destroy the current AR and resume execution in the caller at the statement after the call statement. Each of the three assembly instructions in the answer above contributes to accomplishing the goal.

9. Translate the C++ loop into an equivalent X86 loop [12]

C++:
```
double total = 0.0;
long count = 0;
double cost;
cin >> cost;
do    {total = total + cost;
        count++;
        cin >> cost;
       }while (total < 1000.0);
```

For your information: 1000.0 = 0x408F 4000 0000 0000

Solution is on the next page.

Comment: The colors are an attempt to match parts of the C++ code above with the corresponding parts of the assembly solution on the next page.

There is a correspondence between the C++ and the X86. I am not sure if the color coding helped to illustrate this correspondence.

Solution to question 9.

```
segment .data
floatform db “%lf”,0

  …..

;Block to set up registers for the loop
mov rbx,0
xorpd  xmm15,xmm15
push qword  0x408F400000000000
movsd xmm14, [rsp]
pop rax

;Summary of what we have so far
;xmm14 holds 1000.0
;xmm15 is the accumulator; currently it is zeroed out.
;rbx is the loop counter, currently it holds integer zero

;Block that inputs a single float number from the keyboard into top qword of the stack.
mov rax, 0
mov rdi, floatform
push qword 0
mov rsi,rsp
call scanf
;Special note:  the push will not be reversed until the loop has completed.

begin_loop:
        addsd xmm15,[rsp]
        inc rbx
        ;
        ;Block that inputs one float
        mov rax, 0
        mov rdi, floatform
        mov rsi,rsp
        call scanf
        ;End block for inputting one number
        ;
        addsd xmm15, [rsp]
        ucomisd xmm15, xmm14            ;Compare the accumulator with 1000.0
jb begin_loop
pop rax                 ;Don’t forget to restore the stack to its original state.
```

10. Convert $109\frac{3}{14}$ to 64-bit IEEEE float hex number. [12]

Show sufficient intermediate steps to convince the grader that you know how to do more than operate a calculator.

Solution: First work on the integer part: 109 = 1101101

Next work on the fraction using the technique of repeated multiplication by 2.

(3/14) x 2 = 0 + (3/7)
(3/7) x 2 = 0 + (6/7)
(6/7) x 2 = 1 + (5/7)
(5/7) x 2 = 1 + (3/7)
(3/7) x 2 = 0 + (6/7)
No need to continue because the pattern has begun to repeat.

Therefore, the significant is .00110110110110110110110110110 ……..

Combine the two parts:

Our number = 1101101 . 00110110110110110110110110110 …….. x 2^0

= 1.1011 0100 1101 1011 0110 1101 1011 0110 1101…….. x 2^6

Next add: stored exponent = 3FF + 006 = 405

We are almost done. Combine stored exponent and fractional part into one answer.

Our number = 0x405B 4DB6 DB6D B6DB The end.

11. Convert IEEEE floating point hex number 0x3FC4 8000 0000 0000 to a floating point base 10 decimal number. [11]

Rules: You must show sufficient intermediate steps to convince the grader that you know how to do more than operate a calculator.

Solution: Begin with the stored exponent 3FC. Subtract the bias number.

3FC – 3FF = -3 <== That is negative three

Next convert the mantissa to binary and then to decimal

Significant = 48 followed by many zeros

=1.0100 1000 <==Notice that the hidden 1 bit has been included

Therefore, the original number is 1.0100 1000 x 2^{-3}

The plan is to simplify the significand by moving the point to the right.

Our number is 101001. x 2^{-8}

Use the calculator to find 101001 = 41

Use the calculator to find 2^{8} = 256

Therefore, our number is 41/256 = 0.16015625

12. Convert this twos complement integer 0x2C00 0000 0000 to simplest form base 10 decimal integer. [10]

Rules: You must show sufficient intermediate steps to convince the grader that you know how to do more than operate a calculator.

Solution: The number at first glance looks like an IEEE number, but not in this case. We will apply the rules of integers.

The number is 0010 1100 0000 …. 0000 There are 58 trailing zeros on the right.

The bit on the far right is bit #0

The bit on the far left is bit #63.

Applying the rules of integers our number is

0x(2^63) + 0x(2^62) + 1(2^61) + 0x(2^60) + 1x(2^59) + 1x(2^58) +0 + 0 + 0 +…. + 0

Next drop the trailing zeros and the leading zeros.

Then our number is 1(2^61) + 1x(2^59) + 1x(2^58)

which equals 2^61 + 2^59+ 2^58.. We could stop there, but let's go one step further.

Factor out 2^58 from all three terms.

Our number is (2^58) x [2^3 + 2^1 + 2^0]

Inside the square brackets the number is 8+2+1 = 11.

Conclusion: our number is 11×2^{58} "11 times 2 to the 58"

13. Let the following be a declaration in the segment .data [7]

```
buzz  dq  3.4, 7.9, -8.6, 49.99, -0.3
```

Use a gdb command to output all the numbers of the array in IEEE hex format

Solution preferred: `x/5xg  &buzz`

Alternate solution: `p/x  (double[5])buzz`

14. Suppose good db "Good morning, Sandra",0 is declared in segment .data.

Use a gdb command to output the ascii value in base ten decimal number of each of the first 9 characters of the array good. [7]

Solution: `x/18cb  &good`

There are other solutions; however, the instructor does not know the complete set of all possible solutions.

Alternate solution:
```
x/18db &good
x/17db  (char[99])good
```

There may be others unknown to me. If you gave an answer to question 14, and you have tested the command in a program, and it produces the results required by the question then contact me by email to get your lost points restored.

15. What is the gdb command that will output the low half of xmm13 as one big twos complement signed integer expressed in decimal base ten. [7]

Solution: `p/d  $xmm13.v2_int64[0]`

That's it. Only 15 questions, but the total points for the entire test remains at 100.

Now you have time remaining before 1:30pm. Check your answers. You have the entire internet at your dispose. Use it to verify your answers or improve an answer. If the internet says your answer is all wrong, then consider changing it to blank.

If a question has multiple answers and one of the answers is "blank" then "blank" takes precedence over other answers.

When 3:00pm arrives then send your test document to me as a single file attachment.

Only odt, doc and docx are accepted. Pdf, jpeg, mp3, mkv, etc are not accepted. Don't even think about sending a pdf file.

Mailto: **holliday@fullerton.edu**