**2021 Spring CPSC 240-7**

**May 9, 2021 Concepts Final**

**Time: 9:00am – 2:00pm**

**Cover Sheet**

The official time is 10am-12pm. The extended time is a gesture to make the test a little less stressful for those who can benefit from additional time.

**Only odt or doc or docx**

No later than 2pm send the document with your answers as an attachment to the usual place, namely: holliday@fullerton.edu

**Check answers electronically before submitting**

It is highly recommended that you check your answer online to insure its correctness. If the online calculator says your mathematical answer is wrong then either re-work the solution or choose “blank”. A random wrong answer will (usually) earn less credit than the “blank” answer.

**Grading scale**

Most questions have a single correct answer.

Correct answer ==>100%

“Blank” answer ==> **40%**

Wrong answer ==> 0%

That means: don’t guess random answers. The “blank” option is available to avoid guessing.

**Conclusion**

Everyone in this class is capable of 100% correct answers (except for the matter limited time).

I hope that a year from now you’ll remember Spring 2021 and think “Yes, I did learn some things I didn’t know before.”

**Numeric conversions NC**

Show sufficient steps that will convince the reader that you know the underlying principles involved. It is ok and even recommended to use electronic equipment to verify your answer.

1. Convert 79.7 to IEEE754 hex 64-bit float format.

Show sufficient work to confirm that you know the algorithm.

79 = 1001111

0.7 x2 = 1 + 0.4

0.4 x2 = 0 + 0.8

0.8 x2 = 1 + 0.6

0.6 x2 = 1 + 0.2

0.2 x2 = 0 + 0.4

79.7 = 1001111.1011001100110

1.001111101100 x 2^6

1023+6=1029

1029=10000000101

0100 0000 0101 0011 1110 1100 1100…

0x4053 ECCC CCCC CCCC

2. Convert 0x420B 0000 0000 0000 to base 10 floating point format.

Show sufficient work to confirm you know the algorithm.

Binary: 0100 0010 0000 1011 0000 0000 0000 0000

Determine Positive: 0 first bit = positive

Mantissa = 100 0010 0000 = 1056

1056-1023 = 33

1.1011 0000 0000 0000 0000

11011. 0000 0000 0000 0000 x 2^29

27\*2^29 = 14495514624

14495514624

3. Convert 13.45 x 2-1029 to IEEE754 hex 64-bit float format.

Show sufficient work to confirm you know the algorithm.

13 = 1101

.45x2=0+.9

.9x2=1+.8

.8x2=1+.6

.6x2=1+.2

.2x2=0+.4

.4x2=0+.8

repeat 1100

13.45 = 1101.01110011001100110011

2^-1029 subnormal

1029-1023 = 7

0000 0000 0000 0001 1010 1110 0110 0110...

0x0001 AE66 6666 6666 6666

4. Convert 2/7 to IEEE754 hex 32-bit float format.

Show sufficient work to confirm you know the algorithm.

2/7 = 0.285714….

2/7 x2 = 0+4/7

4/7 x2 = 1+1/7

1/7 x2 = 0+2/7

repeat 010

.0100 1001 0010 0100 1001 0010 0100 x 2^0

1.0010 0100 1001 0010 0100 x2^-2

127-2=125 01111101

0011 1110 1001 0010 0100 1001 0010 0100

0x3E92 4924

5. Convert 0x409A 0000 from IEEE754 32-bit to base 10 float format.

Show sufficient work to confirm you know the algorithm.

0100 0000 1001 1010 0000 0000 0000 0000

1000 0001 = 1+ 128 = 129

129-127 = 2

1.001101000000...x2^2-6= -4

1001101.000 x2 ^-4

77 x 2^-4 = 77 x 0.0625 = 4.8125

4.8125

**Programming = P**

P1. Make an assembly function with this prototype:

double sumofsquares(double arr [ ], long size);

The function returns the sum of all the squares of numbers in the array.

[To save space you should keep the first push rbp but you may omit the remaining 15 pushes. Likewise, you may omit the 15 pops at the end but keep the one pop for rbp.]

global sumofsquares

section .data

giveZero dd 0

section .bss

section .text

sumofsquares:

push rbp

push qword -1

mov r15, rdi

mov r14, rsi

mov r12, 0

movsd xmm11, [giveZero]

begin\_loop:

cmp r12, r14

jge end\_loop

movsd xmm10, [giveZero]

addsd xmm10, [r15 + 8 \* r12]

mulsd xmm10, xmm10

addsd xmm11, xmm10

inc r12

jmp begin\_loop

end\_loop:

movsd xmm0, xmm11

pop rbp

P2. Suppose a large integer such as 750 500 000 000 is stored in r14. You wish to compute 8209/7 of that large integer using only integer arithmetic instructions. Write a fragment of assembly code showing how to do it. Leave the final answer in r15.

“Blank”

P3. Write an assembly block that will read the cpu clock, convert the value read to integer number of seconds (since the machine was booted). Store that resulting number in rbx.

This block will be executed on an Asus Model “Red Dragon” running at max frequency 3.4 GHz. What is the number of seconds?

“Blank”

**Numeric knowledge NK**

NK-1. What is -18230700 in an 9-digit babbage machine?

018230700

0=9

1=8

2=7

3=6

4=5

981769299

+ 1

------------------

981769300

**Legal knowledge**

LK-1. What are two legal responsibilities imposed on a recipient of open source software?

Freedom to run multiple copies of the program on multiple machines

Freedom to physically possess copies of the source code and distribute those copies

LK-2. What is the recommended license to put on software not specific to any one single program?

GNU

LK-3. What is the recommended license to put on a user guide created as part of a group project in CPSC362, and which the professor rated as “Outstanding”?

GNU

**Cultural knowledge**

CK-1. What is considered to be the Opus Magnum of assembly programming?

Roller Coaster Tycoon 2

CK-2. Who is the founder of the movement to promote the use of open source software?

Richard Stallman

CK-3. For what is Grace Hooper known? Give at least one famous action.

She was involved in the creation of COBOL, a high-level programming language that we still use today. She was also apart of making the first fully electronic digital computer named UNIVAC.

**Programmer’s tool box**

PT-1. How do you swap xmm8 and xmm9 without using a temporary register?

xorpd xmm8, xmm9

xorpd xmm9, xmm8

xorpd xmm8, xmm9

PT-2. In a pure assembly program what is the name of the main function?

\_main

PT-3. How do you round a float number in such a way that if the starting number is midway between two integers then the even integer will always win the rounding operation. Show with example.

float num;

int answer;

answer = round(num);

//if answer is odd

if(answer % 2) {

//if answer is greater than original value decrement by 1

if(answer > num) {

answer--;

//if answer is greater than original value increment by 1

} else {

answer++;

}

}

**GDB knowledge**

GDB-1. A dynamic array was created in a C++ program: double arr = new double[8];

Output the first 4 numbers of that array in IEEE754 hex format.

x/4xg arr

GDB-2. Show the low 64-bits of xmm0 in floating point decimal.

p/f $xmm0.v2\_double[0]

GDB-3. Copy the integer -650 into the high 64-bits of xmm3.

set $xmm3.v2\_double[1]=-650

GDB-4. Place these numbers 3.5 7.2 9.4 10.4 into 32-bit positions numbered 0, 1, 2, and 3, respecitively.

“blank”

GDB-5. An array named w has been declared in C++. All the elements of the array are of type double. Show the value in cell #2 of w in IEEE754 hex.

x/xg &w[1]

DCB-6. Show the values in the first 5 cells of w in decimal float numbers.

x/5fg &w

Dcb-7. Show the size (number of quadwords) in the current activation record.

p/d ($rbp-$rsp)/8+1

DCB-8. Change the value on top of the stack to be 5.55.

set var {double}$rsp=5.55

End of test

**Reminder:**

Place your name and email somewhere on this test. This page is a good place for those two data items.

Stefan Parrish

parrishstefan@gmail.com

Send the completed test copy in odt or docx format to the usual place: holliday@fullerton.edu