CS 301 Midterm Exam

0.) NAME: ___Answer Key mk III_____
2016-10-12. Closed book, closed laptop, closed notes.

1.) Assume C++ supports a new "very short int" type that is only 4 bits wide (not bytes, bits!). Show the decimal values corresponding to these bit patterns for "unsigned very short int" and "signed very short int".

binary	0000	0001	0111	1000	1001	1110	1111
unsigned	**0**	**1**	**7**	**8**	**9**	**14**	**15**
signed	**0**	**1**	**7**	**-8**	**-7**	**-2**	**-1**

2.) Here is the 64 bit x86 machine code and disassembly for a tiny function.

```
0:   89 f8                   mov    eax,edi
2:   2d 03 00 00 00          sub    eax,0x3
7:   c3                      ret
```

2.a.) What byte in machine code implements the subtract operation? __0x2d

2.b.) Write the C++ function prototype to allow you to call this function, assuming it is named "fun":

extern "C" int fun(int); // you can tell it's int, because edi and eax

2.c.) Write the equivalent C/C++ function body, assuming the parameter is passed as a variable "bar":

{ return bar-3; }

3.) Fill in the blanks in the table:

Purpose	Assembly	C/C++
Declare a variable:	p: dq 3	long x = 3;
Point at that variable:	mov rdi, __p	long *ptr = ____&x;
Read from the pointer:	mov rcx, __[p]	long y = ____*ptr;

4.) This assembly declaration of a C string prints as "hi!", followed by weird gibberish.

```
db "hi!"
```

How would you fix it? _db "hi!",0__ Why did that happen? ____C string needs a null character terminator (a

zero byte) or the print function will keep printing weird stuff in memory until it hits a zero.

5.) Please fix this code, designed to read and return one array element.

```
; rdi: pointer to int array, rsi: index in array
mov rax, DWORD[rsi + 8*rdi]
ret
```

Ha, there is no fixing it! (target register is wrong size, pointer and index are swapped, scale on index is wrong).

Better to just rewrite from scratch:

```
mov eax, DWORD[rdi + 4*rsi]
ret
```

6.) Given the same rdi and rsi as above, but for a float array, write assembly to return one float array element.

```
vmovss xmm0, [rdi + 4*rsi]
ret
```

7.) Mark each statement as true (T) or false (F) for the 64-bit x86 data types float or int.

Statement	T/F for float?	T/F for int?
Adding two positive numbers can result in a negative number.	F (infinity instead)	T (overflow)
The number of bits used to store a variable depends on its value.	F (always 32 bits)	F (always 32 bits)
All registers are scratch registers.	T (on Linux)	F (rsp, rbp, etc)
There are values x such that: x + 1 == x	T (infinity, or big)	F (low bits different)
There are 16 registers total.	T (xmm0 - 15)	uh, maybe?
Moving a constant 1 into a register is a one-liner.	F (constants in memory only)	T (mov eax,1)

8.) Fill in the blanks to make each line of code return 6, without crashing.

mov rax, 2
add rax, _4
ret
	_push 6
	pop rax
	ret
		sub rsp, 8
		mov QWORD[_rsp_],6
		pop rax
		ret
			sub rsp, 16
			mov QWORD[_rsp+8_],6
			pop rcx
			pop rax
			ret

9.)

When would you use malloc instead of the stack?	When would you use a static dq instead of either one?
1.) When you need to keep the data around after the function returns. 2.) When the data is too big to fit on the stack.	When there is only one copy of the data and you already know the size in bytes, or it's a constant. Static labels can also have names, which is handy.

10.) What db (data byte) values are equivalent to dd 0x12345678 on x86?

db _0x78_, _0x56_, _0x34_, _0x12