CMSC216: Bonus Review 2B

Chris Kauffman

Last Updated:
Sun Nov 2 10:09:36 PM EST 2025

Bonus Review Rules

- 3 Questions will be shown with about 5min per question, 15min total, time limit enforced on Gradescope Quiz
- Individual student bonus dots will be calculated as

```
BonusDots = floor(log2(TotalCorrectSectionAnswers))
            - YourIncorrectAnswers
```

- Cooperation is allowed and encouraged within your discussion section: the more correct answers in the section, the more bonus points for all
- Staff will try to facilitate discussion but will not comment on correct/incorrect answers during the quiz
- Scores will posted after all sections have taken the done the bonus review, likely the following day
- Student in the Discussion Section with the highest `TotalCorrectSectionAnswers` will get +2 BonusDots
- Bonus Review is Open Resource just like the exam: https://www.cs.umd.edu/~profk/216/exam-rules.pdf

Staging

- Open up the Gradescope Bonus Review Quiz for the day
- Once started, the quiz closes after 15min
- Get your resources set for the quiz

Okay...

Question 1

```
1  .text
2  .global minor_test_of_strength
3  minor_test_of_strength:
4    movq $1, %rax
5    movq $1, %rcx
6    testl %eax, %ecx
7    je .RET_1
8    movq $0, %rax
9    ret
10 .RET_1:
11   ret
```

This function will return...

- (A) 1 every time
- (B) 0 every time
- (C) Either 1 or 0 based on its parameter(s) which determine the function behavior
- (D) Either 1 or 0 but it's impossible to say which since the setup and use of the `testl` instruction leads to undefined behavior
- (E) A really large and unpredictable number since moving $1 into `%rax` causes an overflow
- (F) Trick Question: it won't compile as the `testl` instruction uses one register, not two

Question 2

```
.text
.global add2strs
add2strs:  ## Has bugs...
   subq    $16, %rsp
   movq    %rcx, %rbp
   call    convert
   movl    %edi, %ebx
   movq    %rbp, %rdx
   call    convert
   addl    %ebx, %edi
   addq    $16, %rsp
   ret
```

```
1  int convert(char *str);
2
3  // Add 2 strings together by converting them
4  // to integers and summing those integers
5  int add2strs(char *astr, char *bstr){
6    int anum = convert(astr);
7    int bnum = convert(bstr);
8    int sum = anum+bnum;
9    return sum;
10 }
```

Above is a correct C implementation of `add2strs()`. Which of the following is NOT A MISTAKE in the Assembly implementation of the `add2strs()` function?

- (A) Does not align the stack correctly for a function call
- (B) Uses the wrong register for First function parameter
- (C) Uses the wrong register for Second function parameter
- (D) Uses Caller-save registers without saving/restoring them
- (E) Uses Callee-save registers without saving/restoring them
- (F) Uses the wrong register for return values

Question 3

One of these things is not like the others,
One of these things just doesn't belong,
Can you tell which thing is not like the others
By the time I finish my song?

- (A) Program Counter
- (B) PC
- (C) Instruction Pointer
- (D) EFLAGS
- (E) RIP
- (F) Address of next instruction