! This class has been made inactive. No posts will be allowed until an instructor reactivates the class.

note @906 142 views

Actions

[Past Midterms] 2018 and older

You can find the past exams here: https://cs61c.org/sp22/resources/exams/

When posting questions, please reference the semester, exam, and question in this format so it's easier for students and staff to search for similar questions:

Semester-Exam-Question Number

For example: **SP18-MT1-Q1**, or **FA17-MT2-Q3**

exam exam/midterm

good note 0

Updated 5 months ago by Jerry Xu and Caroline Liu

followup discussions, *for lingering questions and comments*

[✓ Resolved] Unresolved @906_f1

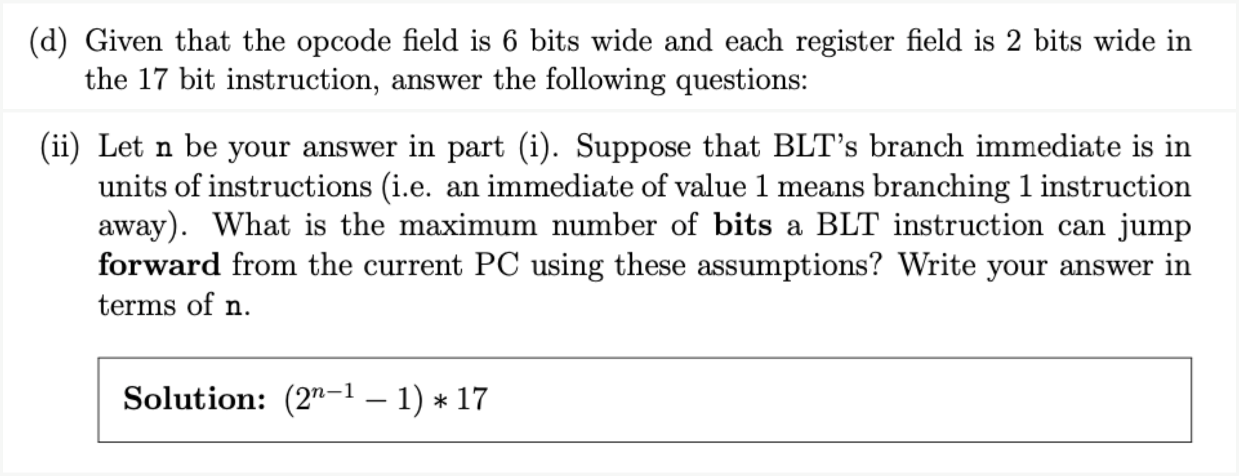
This was marked a duplicate to the question/note above by Peyrin Kao 7 months ago

Anonymous Beaker 7 months ago

sp18-mt1 questions

[sp18-mt1-q5.d.2]

Why do we need to multiply a 17 here?



(d) Given that the opcode field is 6 bits wide and each register field is 2 bits wide in the 17 bit instruction, answer the following questions:

(ii) Let **n** be your answer in part (i). Suppose that BLT's branch immediate is in units of instructions (i.e. an immediate of value 1 means branching 1 instruction away). What is the maximum number of **bits** a BLT instruction can jump **forward** from the current PC using these assumptions? Write your answer in terms of **n**.

Solution: $(2^{n-1} - 1) * 17$

helpful! 0

Akshat Jain 7 months ago

In this problem, each instruction is 17 bits. so instructions aren't 4 bytes away from each other, they're 17 bits away.

The immediate is n bits, and it's a signed two's complement number since we want to be able to jump forward (positive imm) and backward (negative imm). For 2's comp numbers with n bits, the range of possible values is $[-2^{n-1}, 2^{n-1} - 1]$.

Remember that this immediate determines how much we jump by.

Since the question is asking for the maximum number of bits a *forward* jump can happen, we take the most *positive* value for the jump ($2^{n-1} - 1$), and we multiply by 17 because each instruction is 17 bits.

good comment | 0

Anonymous Beaker 7 months ago

If we are asking about the number of instructions instead, would the answer become simply 2^(n-1) - 1?

helpful! | 0

Akshat Jain 7 months ago

Yes!

good comment | 0

Reply to this followup discussion

[✓ Resolved] Unresolved @906_f2

This was marked a duplicate to the question/note above by Peyrin Kao 7 months ago

Anonymous Comp 7 months ago

Fall 2017 midterm questions

Q3.1: is it wrong if we cast the malloc statement with (struct list_node *)?

Q3.2: why is there a memory leak?

Q4.1: What's the difference between B and D, and why is D incorrect?

Q4.3: is the reason why we need an if statement for (node->word) because strlen does not work on empty character arrays?

Thank you so much!

helpful! | 0

Akshat Jain 7 months ago

Q3.1: is it wrong if we cast the malloc statement with (struct list_node *)?

> *Nope that's totally fine! C will automatically cast the `void *` pointer that `malloc` returns, but explicitly defining the type is never a bad idea!*

Q3.2: why is there a memory leak?

> *In test_reverse we are malloc'ing a `head` variable. But in reverse, we are changing what that `head` variable was pointing to. Thus, we lose access to the original memory address and we can't free it later.*

Q4.1: What's the difference between B and D, and why is D incorrect?

> *D won't compile. There's a syntax error in the multi-variable pointer declaration for the `struct TSTnode` fields.*

Q4.3: is the reason why we need an if statement for (node->word) because strlen does not work on empty character arrays?

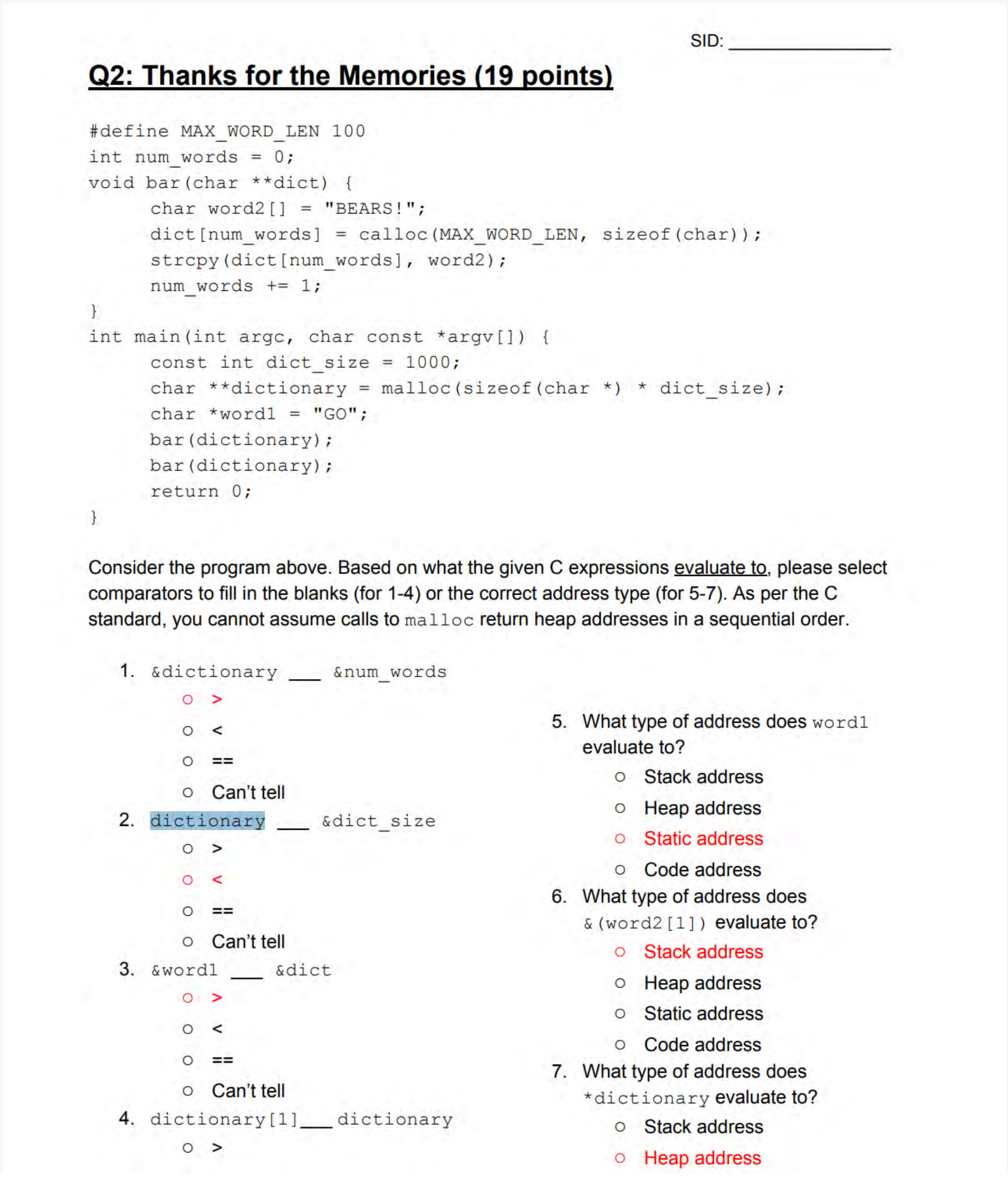
The if statement is checking if `node->word` even exists. To do this, it needs to make sure it isn't NULl. If it is NULL, the strlen function call will segfault.

good comment | 0

Reply to this followup discussion

[✓ Resolved] Unresolved @906_f3

Anonymous Scale 7 months ago



SID: ________________

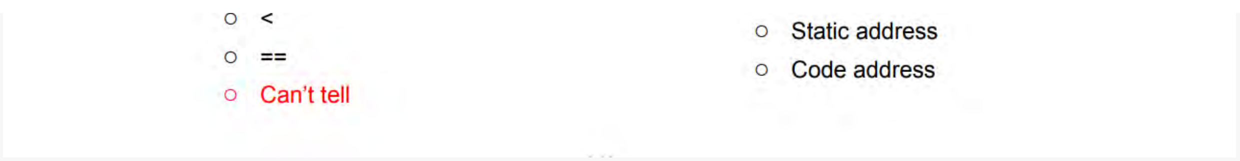
Q2: Thanks for the Memories (19 points)

```
#define MAX_WORD_LEN 100
int num_words = 0;
void bar(char **dict) {
      char word2[] = "BEARS!";
      dict[num_words] = calloc(MAX_WORD_LEN, sizeof(char));
      strcpy(dict[num_words], word2);
      num_words += 1;
}
int main(int argc, char const *argv[]) {
      const int dict_size = 1000;
      char **dictionary = malloc(sizeof(char *) * dict_size);
      char *word1 = "GO";
      bar(dictionary);
      bar(dictionary);
      return 0;
}
```

Consider the program above. Based on what the given C expressions evaluate to, please select comparators to fill in the blanks (for 1-4) or the correct address type (for 5-7). As per the C standard, you cannot assume calls to `malloc` return heap addresses in a sequential order.

1. `&dictionary ___ &num_words`
   - [✓ >]
   - <
   - ==
   - Can't tell
2. `dictionary ___ &dict_size`
   - >
   - [✓ <]
   - ==
   - Can't tell
3. `&word1 ___ &dict`
   - [✓ >]
   - <
   - ==
   - Can't tell
4. `dictionary[1]___ dictionary`
   - >

5. What type of address does `word1` evaluate to?
   - Stack address
   - Heap address
   - [✓ Static address]
   - Code address
6. What type of address does `&(word2[1])` evaluate to?
   - [✓ Stack address]
   - Heap address
   - Static address
   - Code address
7. What type of address does `*dictionary` evaluate to?
   - Stack address
   - [✓ Heap address]



- <
- ==
- Can't tell

- Static address
- Code address

For part 2 what will dictionary evaluate to?

Will it be something in the stack or the heap?

Thanks!

helpful! | 0

Jero Wang 7 months ago

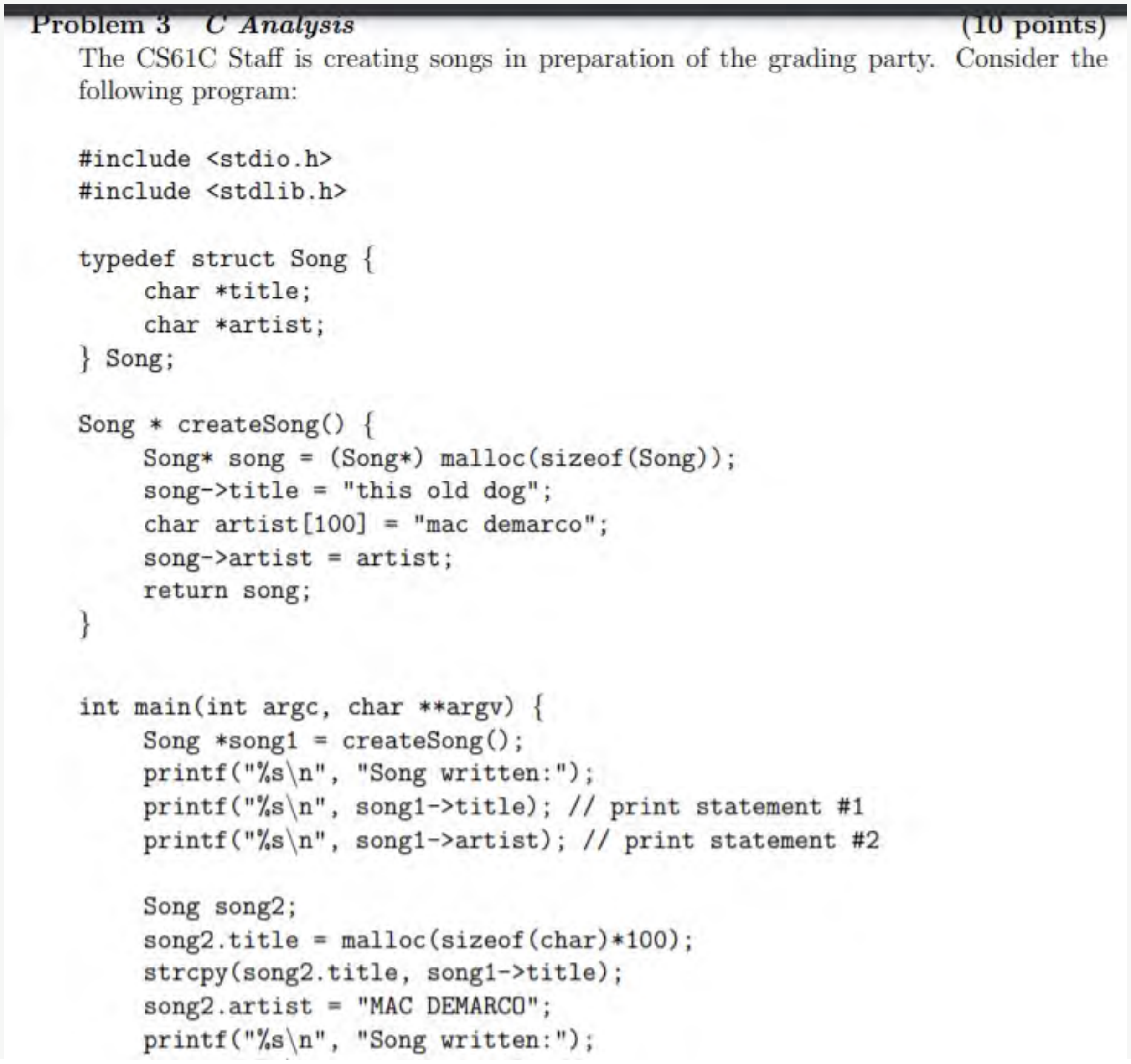
`dictionary` evaluates to an address on the heap, since it stores the return value from a `malloc` call. `dict_size` is stored on the stack, so `&dict_size` is a stack address. Heap will always have lower addresses than the stack, so the answer is `<` .

good comment | 0

Reply to this followup discussion

[✓ Resolved] Unresolved @906_f4

Anonymous Atom 7 months ago



Problem 3 *C Analysis* (10 points)

The CS61C Staff is creating songs in preparation of the grading party. Consider the following program:

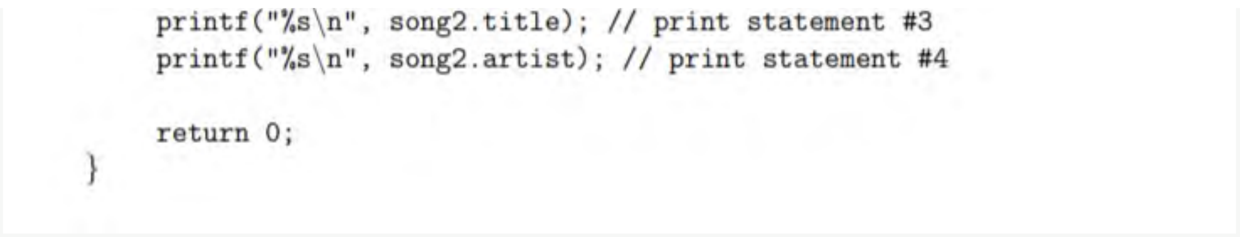
```
#include <stdio.h>
#include <stdlib.h>

typedef struct Song {
     char *title;
     char *artist;
} Song;

Song * createSong() {
     Song* song = (Song*) malloc(sizeof(Song));
     song->title = "this old dog";
     char artist[100] = "mac demarco";
     song->artist = artist;
     return song;
}


int main(int argc, char **argv) {
     Song *song1 = createSong();
     printf("%s\n", "Song written:");
     printf("%s\n", song1->title); // print statement #1
     printf("%s\n", song1->artist); // print statement #2

     Song song2;
     song2.title = malloc(sizeof(char)*100);
     strcpy(song2.title, song1->title);
     song2.artist = "MAC DEMARCO";
     printf("%s\n", "Song written:");
```



```
        printf("%s\n", song2.title); // print statement #3
        printf("%s\n", song2.artist); // print statement #4

        return 0;
    }
```

For Q3 of Spring 2018 MT1, I have a question outside of the ones asked to get a better idea of the concepts. If we were asked to find out what type of address &song1 would evaluate to, what would it be and why?

helpful! | 0

Adelson Chua 7 months ago

song1 is a pointer to a structure Song which was returned by createSong() function.

&song1 will be a pointer to that pointer. It will be some address (we don't know what) in the memory that points to the song1 pointer. It will be of type Song**

good comment | 0

Arnav Jain 7 months ago

Would song1 be in heap memory?

helpful! | 0

Adelson Chua 7 months ago

song1 as a variable is in the stack (within main), but it is pointing to the structure that is in the heap (due to the malloc call initialization at createSong())

good comment | 0

Giulianna Hashemi-Asasi 7 months ago

On this same question, for statement 2 to print the string, should we set song -> artist = * artist instead?

Also, what exactly is the difference between the arrow notation and dot notation? Thank you in advance!

helpful! | 0

Adelson Chua 7 months ago

The difference between arrow and dot should've been covered in the lecture...

If you have a pointer to a struct, *song1, you access the element by (*song1).title. This is equivalent to song1->title.

The arrow notation is a shorthand for dereferencing the struct point and using the dot notation to access the element.

good comment | 0

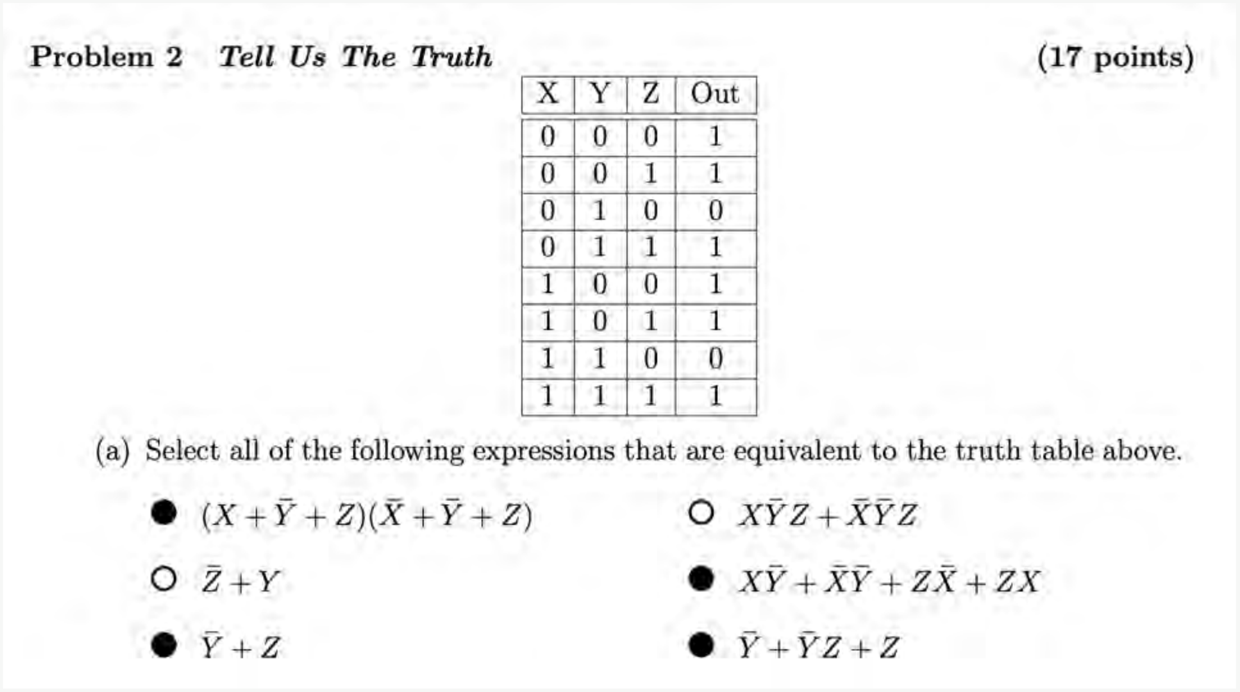
Reply to this followup discussion

[✓ Resolved] Unresolved @906_f5

Anonymous Beaker 7 months ago

[sp18-mt2-q2a]

How to get option 1 (top left) in the solution?



Problem 2 ***Tell Us The Truth*** **(17 points)**

X	Y	Z	Out
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

(a) Select all of the following expressions that are equivalent to the truth table above.

● $(X+\bar{Y}+Z)(\bar{X}+\bar{Y}+Z)$ ○ $X\bar{Y}Z+\bar{X}\bar{Y}Z$

○ $\bar{Z}+Y$ ● $X\bar{Y}+\bar{X}\bar{Y}+Z\bar{X}+ZX$

● $\bar{Y}+Z$ ● $\bar{Y}+\bar{Y}Z+Z$

helpful! | 0

Adelson Chua 7 months ago

There is a technique to do this, but it was not covered in the lecture, so I don't want to introduce a new method for you. Unless you insist.

Alternatively, since the choices are given, what you can do is just evaluate the expression and check if it matches the truth table that is given. That is, substitute X = 0, Y= 0, Z=0, then evaluate if it outputs a 1, and so on.

good comment | 0

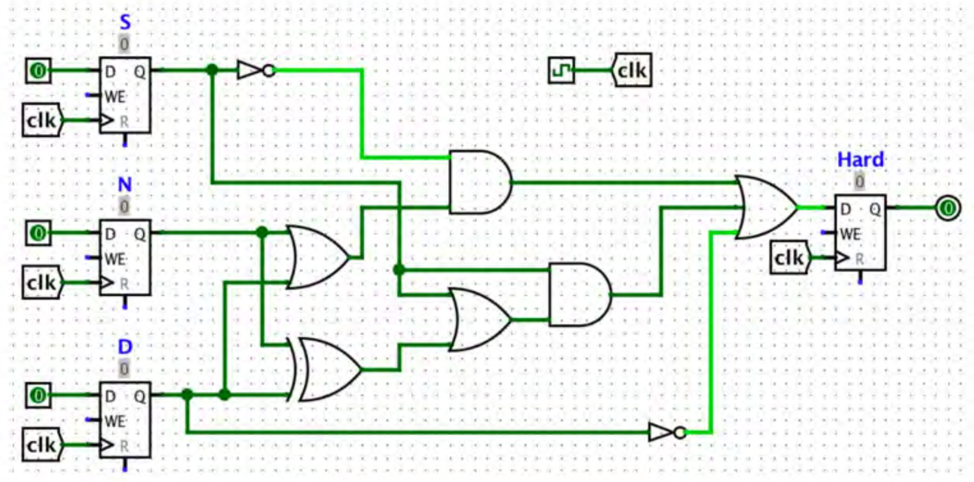
Reply to this followup discussion

[✓ Resolved] Unresolved @906_f6

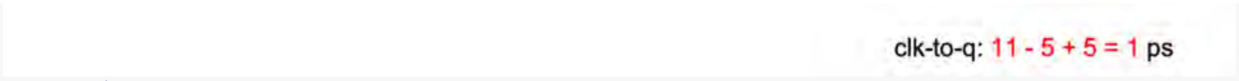
Anonymous Beaker 7 months ago

[su18-mt2-q2]

How to get 11-5+5 = 1??



Using the assumptions above, what is the smallest value that the clk-to-q can be and **not** cause a hold-time violation?


clk-to-q: 11 - 5 + 5 = 1 ps

helpful! | 0

Adelson Chua 7 months ago

There's a typo, it should've been 11-(5+5) = 1.

Remember the inequality, thold >= tshortest, and tshortest is the shortest logic path + tclk-to-q.

11 >= tclk-to-q + (5 + 5), the shortest path is from the D register output to the Hard register output, so 2 gate delays.

11-(5+5) >= tclk-to-q. Then were looking for the smallest, so that's the point where they are equal.

Thus, 11-(5+5) = 1 = tclk-to-q

good comment | 0

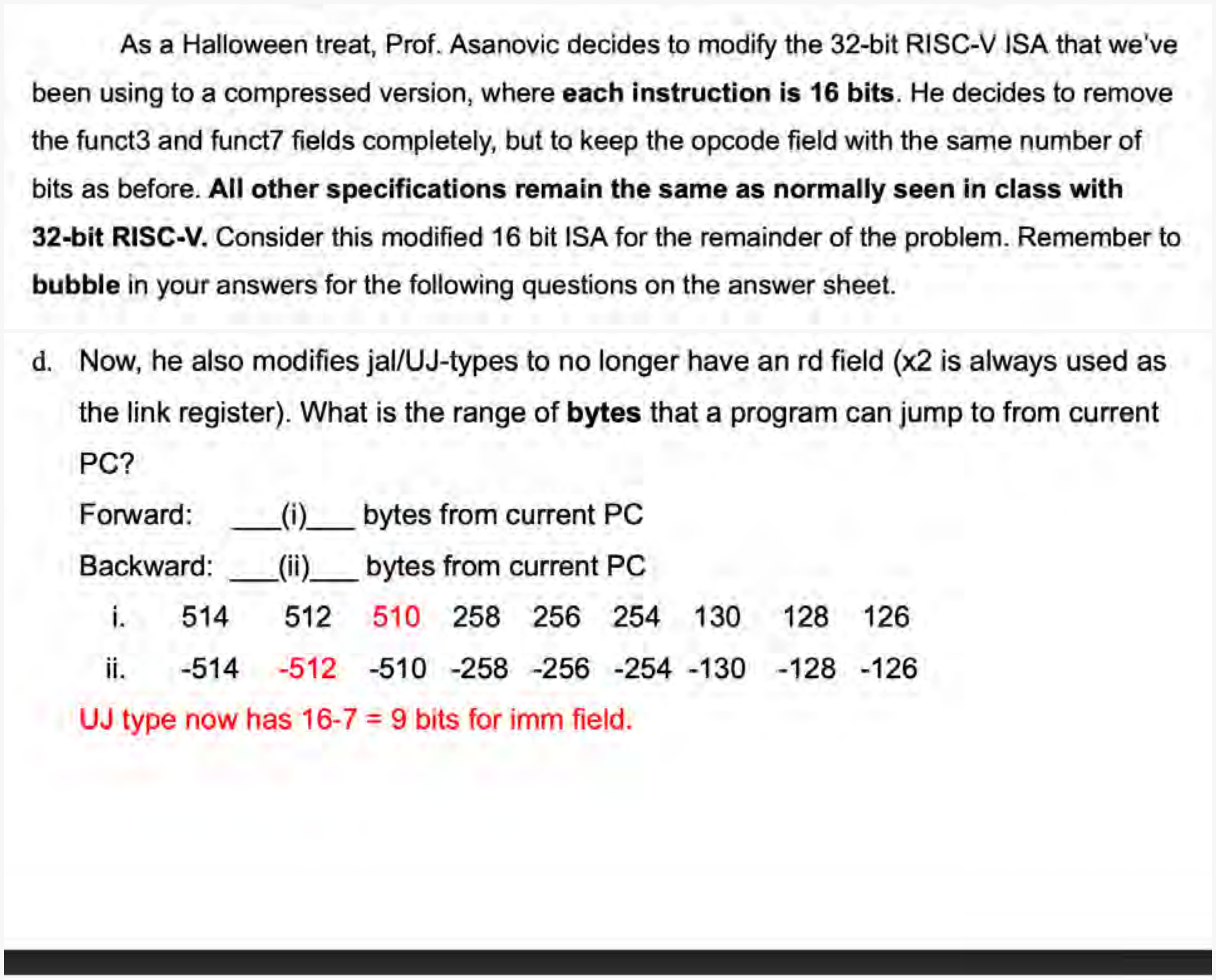
Reply to this followup discussion

[✓ Resolved] Unresolved @906_f7

Anonymous Beaker 7 months ago

[fa17-mt2-q1.d]

How to calculate how many bytes we can jump and also why there's a *2 at the end?


As a Halloween treat, Prof. Asanovic decides to modify the 32-bit RISC-V ISA that we've been using to a compressed version, where **each instruction is 16 bits**. He decides to remove the funct3 and funct7 fields completely, but to keep the opcode field with the same number of bits as before. **All other specifications remain the same as normally seen in class with 32-bit RISC-V.** Consider this modified 16 bit ISA for the remainder of the problem. Remember to **bubble** in your answers for the following questions on the answer sheet.

d. Now, he also modifies jal/UJ-types to no longer have an rd field (x2 is always used as the link register). What is the range of **bytes** that a program can jump to from current PC?

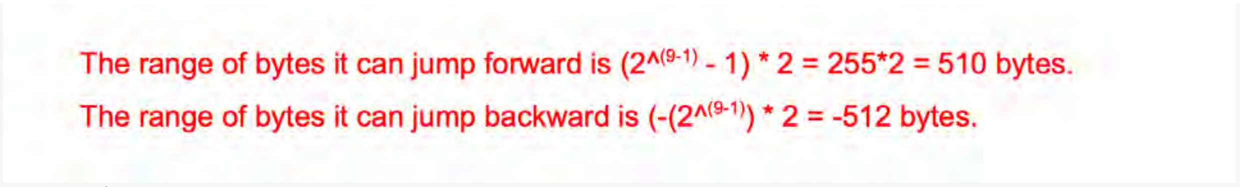
Forward: ___(i)___ bytes from current PC

Backward: ___(ii)___ bytes from current PC

i. 514 512 [✓ 510] 258 256 254 130 128 126

ii. -514 [✓ -512] -510 -258 -256 -254 -130 -128 -126

UJ type now has 16-7 = 9 bits for imm field.



The range of bytes it can jump forward is $(2^{(9-1)} - 1) * 2 = 255*2 = 510$ bytes.

The range of bytes it can jump backward is $(-(2^{(9-1)}) * 2 = -512$ bytes.

helpful! | 0

Rosalie Fang 7 months ago

First, the number of bytes it can jump depends on how many bits are usable to represent address. In this question, because the instruction will be made up of 7 bits of opcode and 16 bits total, the immediate field will have 9 bits. The reason for the *2 at the end is that, remember, for UJ type instructions, the least significant bit is assumed to be 0 and not included in the imm field. So the actual range of bytes forward will be (first bit is sign bit) 0b01 1111 1110, where only the most significant 9 bits are actually written in the imm field, and similarly for backward range.

good comment | 0

Anonymous Beaker 7 months ago

Why do we need 9-1 as the exponent?

helpful! | 0

Anonymous Beaker 7 months ago

Ohhh is it because it's basically 2's complement range?

helpful! | 0

Adelson Chua 7 months ago

Yes

good comment | 0

Reply to this followup discussion

Start a new followup discussion

Compose a new followup discussion