ECE 2400 Computer Systems Programming
Topic 4: C Pointers

`http://www.csl.cornell.edu/courses/ece2400`
School of Electrical and Computer Engineering
Cornell University

revision: 2021-10-14-11-33

Please do not ask for solutions. Students should compare their solutions to solutions from their fellow students, discuss their solutions with the instructors during lab/office hours, and/or post their solutions on Ed for discussion.

List of Problems

Problem 1. Short Answer

Carefully plan your solution before starting to write your response. Please be brief and to the point; if at all possible, limit your answers to the space provided.

Part 1.A Working with Lines

The following `calc_slope` function calculates the slope of a 2D line, and the `translate_x` function translates a 2D line in the X direction by a given shift amount. **Draw the state diagram that corresponds to the execution of this C program.** You must clearly label all variables in your diagram.

```
// User-defined types for points and lines

typedef struct { int x; int y; } point_t;
typedef struct { point_t pt0; point_t pt1; } line_t;

// Function for calculating the slope

float calc_slope( line_t line )
{
  float rise = line.pt1.y - line.pt0.y;
  float run  = line.pt1.x - line.pt0.x;
  return rise / run;
}

// Function for translating a line

void translate_x( line_t* line_p, int shift )
{
  line_p->pt0.x += shift;
  line_p->pt1.x += shift;
}

// Main function

int main( void )
{
  line_t line;
  line.pt0.x = 1;
  line.pt0.y = 1;
  line.pt1.x = 2;
  line.pt1.y = 3;

  float slope = calc_slope( line );
  translate_x( &line, 4 );

  return 0;
}
```

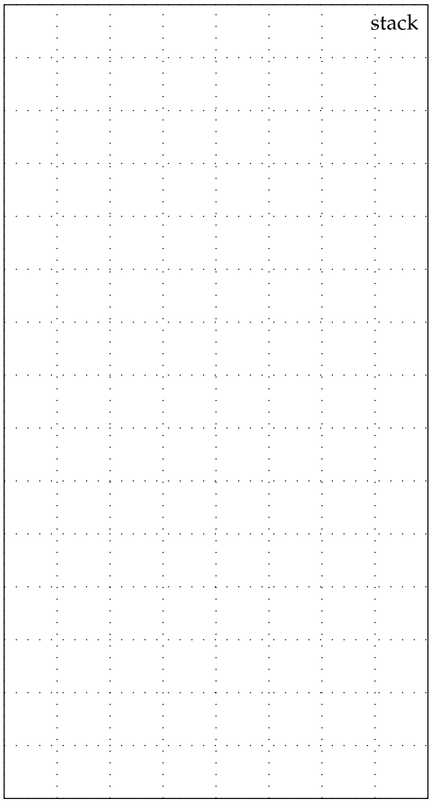
stack

Part 1.B Conceptual Storage vs. Machine Memory for Pointers

The following code snippet illustrates using pointers. **Draw the conceptual state diagram that corresponds to the execution of this C program.** Clearly label all variables.

Once you have finished the conceptual state diagram, **draw the machine memory diagram that also corresponds to the execution of this C program.** Assume that the machine only has a total of 128 bytes of memory. Clearly label the location of each variable in memory. We have already allocated the variable a to get you started. Recall that a variable of type `int` is 32 bits (four bytes), and that we always arrange variables such that the least significant ("right most") byte is at the lowest address in memory. Assume that pointers are also 32 bits (four bytes). You do not need to show values in machine memory in base two. Each byte can be a decimal number. You do not need to show how code maps into machine memory.

```
int   a    = 42;
int   b    = 13;
int*  ptr0 = &a;
int*  ptr1 = ptr0;
int** ptr2 = &ptr1;
int   c    = *ptr0 + **ptr2;
```



Memory (byte addr)

127	0	a
126	0	
125	0	
124	42	
123		
122		
121		
120		
119		
118		
117		
116		
115		
114		
113		
112		
111		
110		
109		
108		
107		
106		
105		
104		
...		
...		
3		
2		
1		
0		