ECE 220 HKN Midterm 2 Review

Aditya Chitnis
Saransh Sinha
Vyom Thakkar

Memory

Address	Value
0xECEB	0x15
0xECEC	0xFE
0xECED	0x1A
0xECEE	0x53

- Memory can be thought of as separated into two parts
  - Address
    - Answers the question of WHERE the data is
  - Value
    - Answers the question of WHAT the data is
- You need to know both parts in order to manipulate data or do anything useful
- EVERYTHING is stored in memory

Pointers

- * = dereference operator & = reference (“address-of”) operator
- Example: swap two numbers
  - ```
    void swap(int * a, int * b){
        int t = *a;  // save the value of a into the temp variable t

        *a = *b;     // save the value of b in the memory location pointed to
                     // by a
        *b = t;      // save the temp value in the memory location pointed to
                     // by b
    }
    ```
  - ```
    int a = 5;
    int b = 3;
    swap(&a, &b);
    ```

Pointers Pt. 2 - Arrays and Strings

- An array is simply a collection of elements found sequentially in memory
  -

Index	0	1	2	3	4	5	6	7	8	9
Value	?	?	?	?	?	?	?	?	?	?

  - Declare the above array
    - `int arr[10]; // an array of 10 elements on stack`
- Strings are arrays of characters that are NULL terminated
  - `char * str = "ECE 220";`

'E'	'C'	'E'	' '	'2'	'2'	'0'	'\0'

- 2D Arrays
  - `int arr_2d[5][5]; // access directly`
    `int arr_2d[25]; // row * num_cols + cols`

Structs

- Structs are a way to aggregate several characteristics of a real world object and treat them as one entity.
  - Ex.

```
typedef struct book_t{          // typedef not needed, only if you
    int isbn;                   // want
    char * author;
    char * title;
    char * text;
}  book;
```

  - Variable Declaration:

```
struct book understandingTheLinuxKernel;
```

  - Accessing a member variable:

```
understandingTheLinuxKernel.author;
```

- ***Note***: The total memory a struct occupies is the total of the memory occupied by each member

Recursion & Backtracking

- General Form of A Recursive Function:
  - ```
    int rec_func(){
        /* check if at base case */
        if(base_case){return 0;}
        /* function logic */
        … // [insert logic here]
    }
    ```
  - Generally involves using a solution to a more basic problem and a reductive step to
  - *Can* function similarly to a loop
- Backtracking
  - The act of checking if the recursive function is valid and can return a concrete answer
    - If not, undo the last call and backtrack
    ```

Recursion Tips

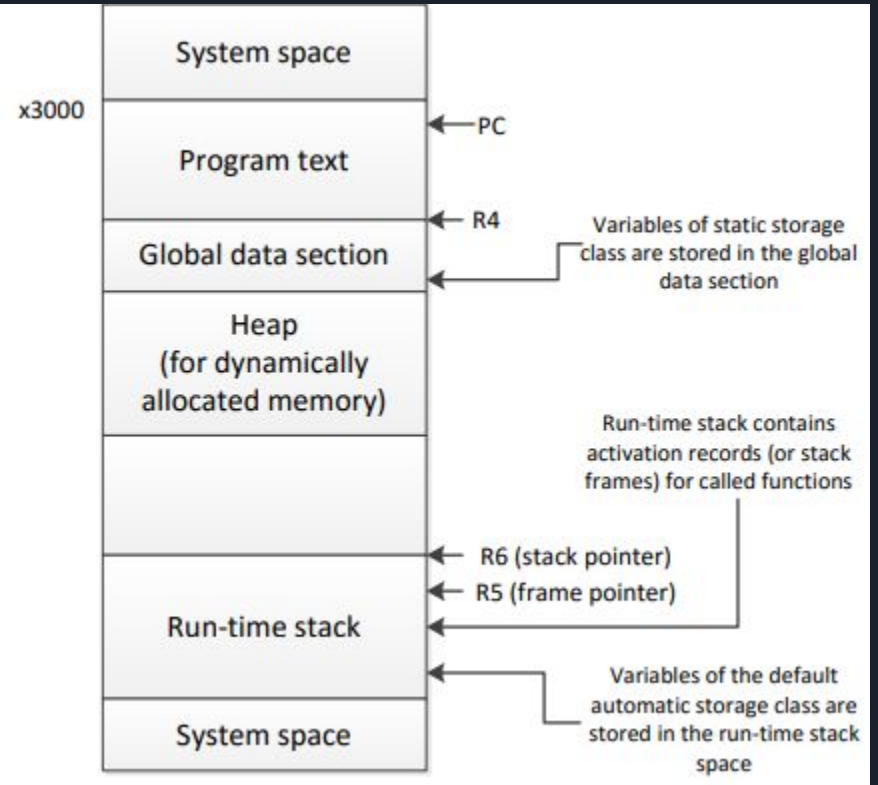
- Doing Recursion --- EASY ; Base Case + Reduction (recursive step)
- Convincing yourself that recursion works --- SOMETIMES HARD (domino example)
- When to use recursion? --- The solution exists in a naturally recursive form ; Eg. factorial, fibonacci, trees (naturally recursive data structure)

Backtracking Example (N Queens) and General Template

```
N-Queens( board[][], N )
    if N is 0                          //All queens have been placed
        return true
    for i = 1 to N {
        for j = 1 to N {
            if is_attacked(i, j, board, N) is true
                skip it and move to next cell
            board[i][j] = 1             //Place current queen at cell (i,j)
            if N-Queens( board, N-1) is true     // Solve subproblem
                return true                      // if solution is found return true
            board[i][j] = 0             /* if solution is not found undo whatever changes
                                          were made i.e., remove  current queen from (i,j)*/
        }
    }
    return false
```

MP 9 (Maze) and Lab 9 (Vectors) should give you a good feel on bactracking strategy

Run-Time Stack



Important Registers:

- R4 : Global Data Section
- R5 : Base of runtime stack
- R6: Top of runtime stack
- R7: Return Address

Updating the Runtime Stack:

Push:

```
ADD R6, R6, #-1        ;Update pointer
STR R0, R6, #0         ;Push data in
R0
```

Pop:

```
ADD R6, R6, #-1        ; Update
pointer
```

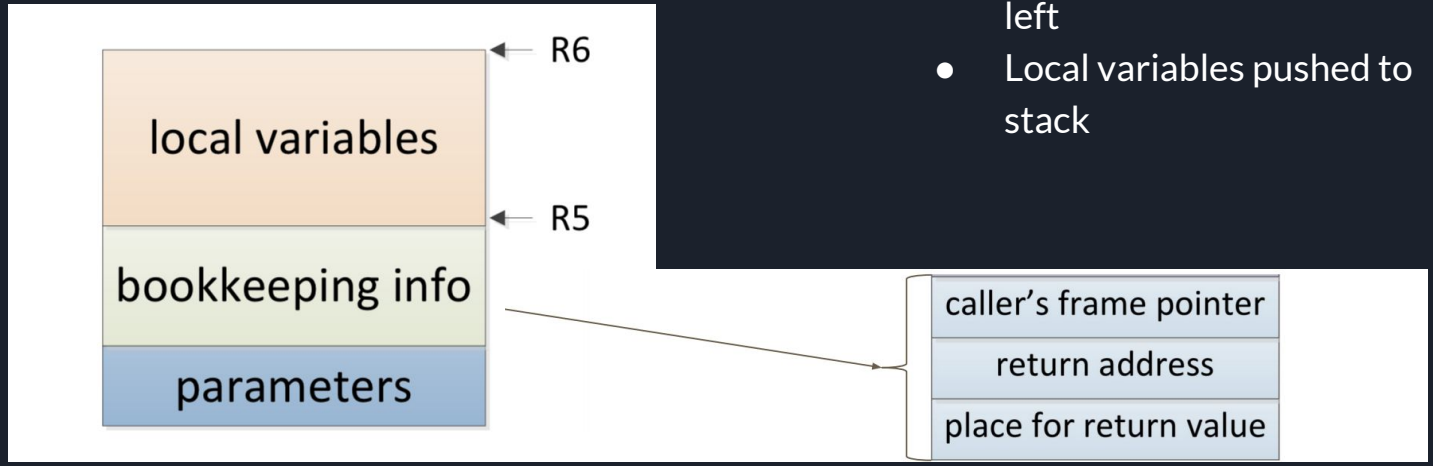
Run-Time Stack (Continued)

Bookkeeping info:

- Callee frame pointer
- Return address
- Return value

Things to Remember:

- Activation Record: how to create and tear-down
- The record is popped when exiting the function
- Update the stack frame
- Parameters pushed right to left
- Local variables pushed to stack



Practice Time!

https://codeshare.io/5XlJ7l