

15-213 Recitation 5

Introduction to Computer Systems

Fahim Dalvi

26 September, 2013

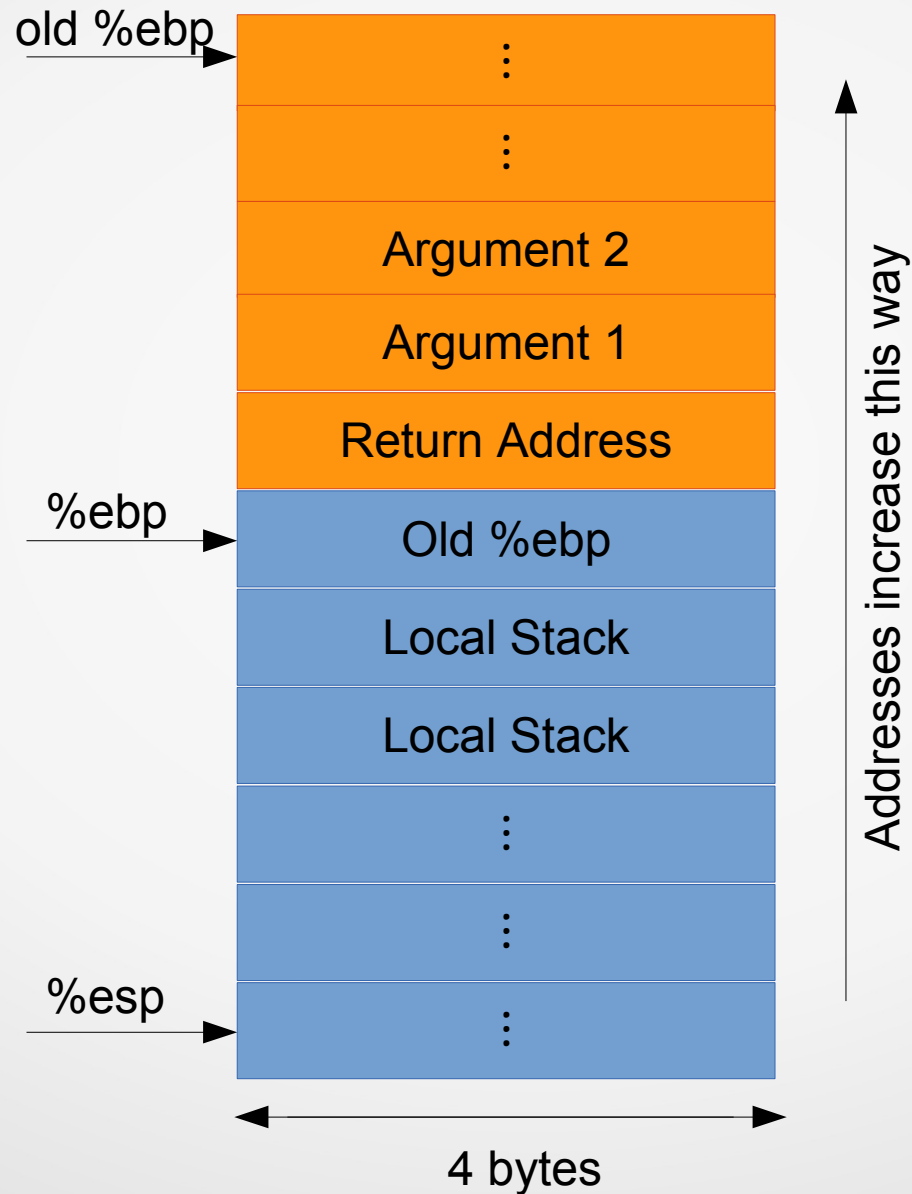
Today

- Buflab
- Arrays
- Structs
 - Data Alignment
- Unions

Buflab

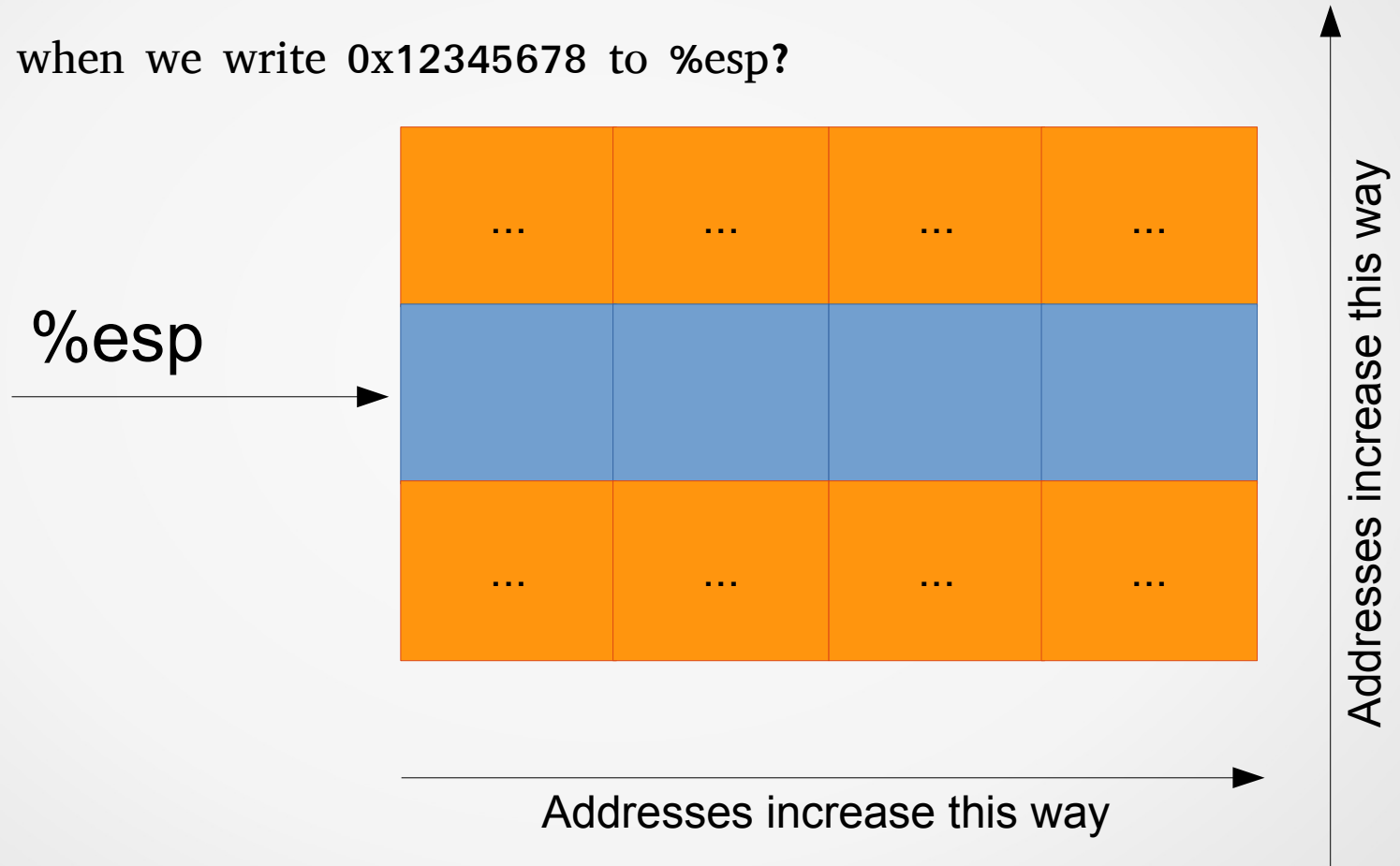
- Due: Wednesday, 2nd October
- Not a lot of time for this lab
- A series of exercises asking you to overflow the stack and change execution
 - You do this by providing inputs that are long, really long
 - No negative grading \o/

Buflab – Remember the stack?



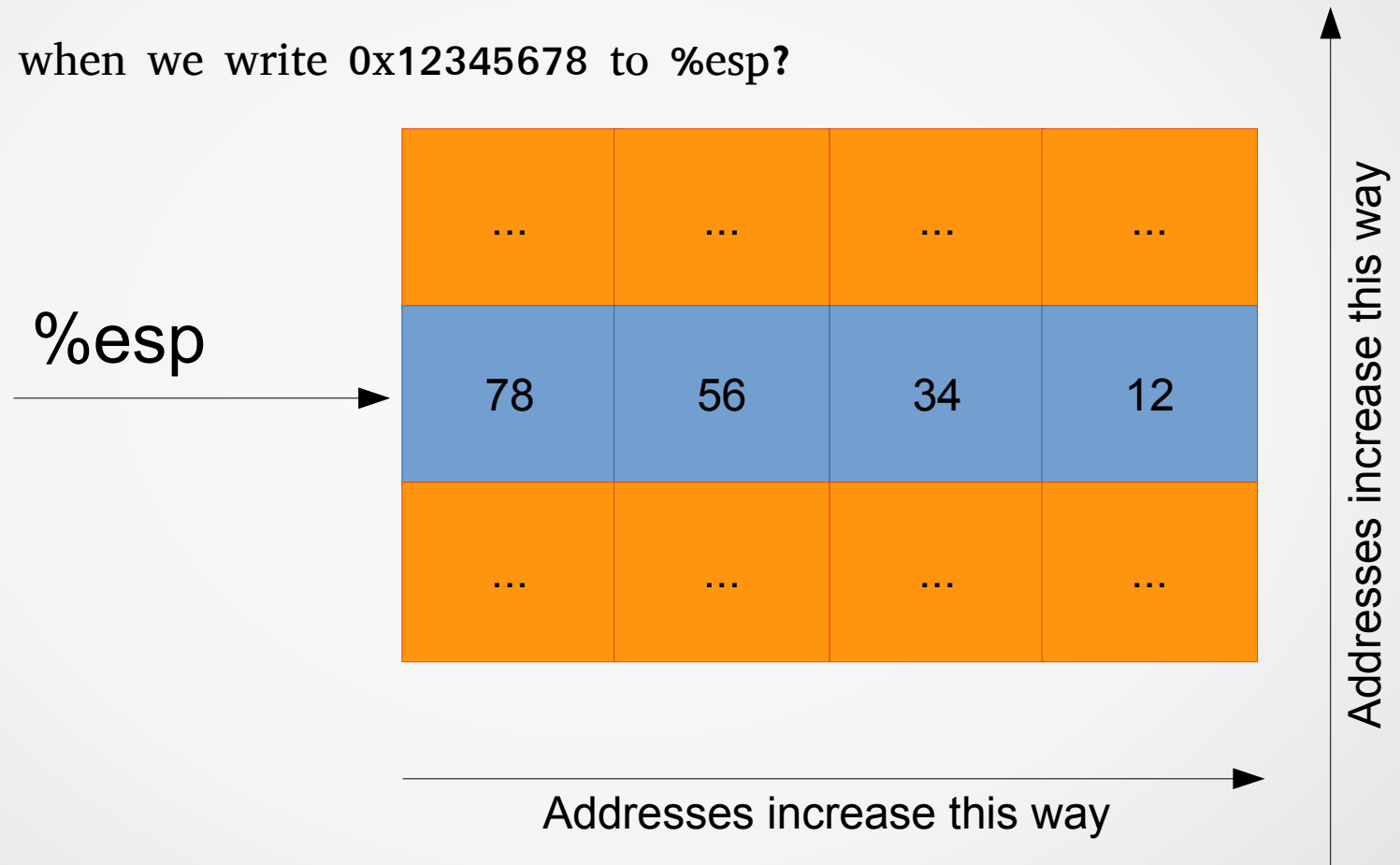
Buflab – Byte ordering

What happens when we write 0x12345678 to %esp?



Buflab – Byte ordering

What happens when we write 0x12345678 to %esp?



Buflab - Exploits

- Earlier stages:
 - Put your byte code in a file
 - Feed it through hex2raw
- Later stages:
 - Write the required “corruption” code in a C file
 - Compile it
 - > `gcc -m32 codeFile.c`
 - Disassemble it
 - Use the hex codes there to create your exploit strings

Buflab

- Start early
- Read the writeup
 - Very detailed
 - Gives excellent hints
- Feel free to ask questions on Piazza!

Arrays

- You've hopefully seen them in Bomblab
- Just contiguous memory locations
- Multidimensional Arrays
 - Stored in row order
- **Remember: No bound checking is done!**
- Hence accessing the 100th location in an array of length 90 would give non-deterministic errors

Struct

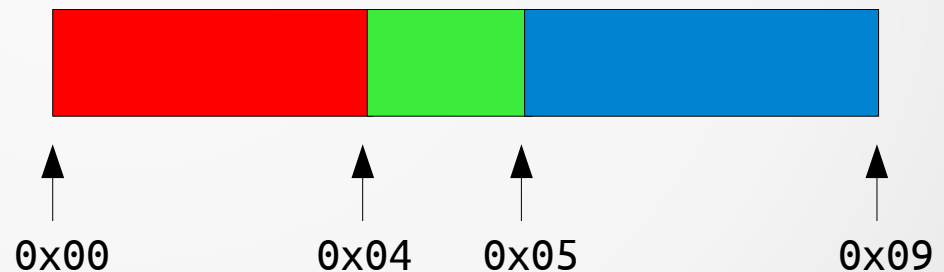
- Structs are also stored as contiguous memory
- Lets look at an example:

```
struct someStruct{  
    int someInt;  
    char someChar;  
    int someInt;  
}
```

Struct

- Structs are also stored as contiguous memory
- Lets look at an example:

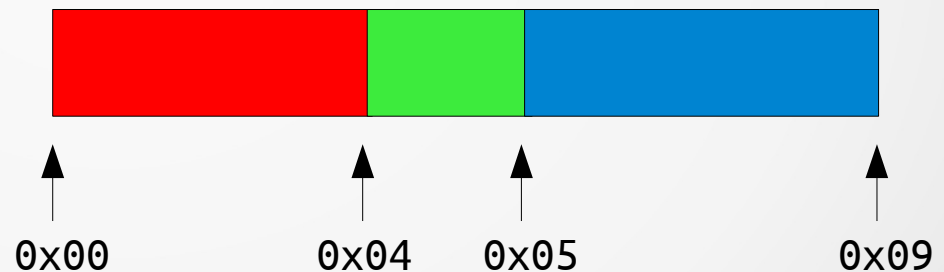
```
struct someStruct{  
    int someInt;  
    char someChar;  
    int someInt;  
}
```



Struct

- Structs are also stored as contiguous memory
- Lets look at an example:

```
struct someStruct{  
    int someInt;  
    char someChar;  
    int someInt;  
}
```

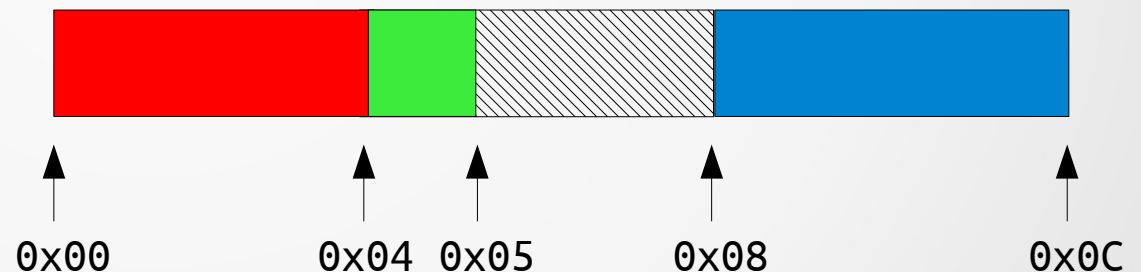


But wait, are we
forgetting something?

Struct – Data Alignment

- We need to align the individual elements
- Lets look at an example:

```
struct someStruct{  
    int someInt;  
    char someChar;  
    int someInt;  
}
```



Data Alignment – Quick Cheat Sheat

Type	x86	x86-64
char	-	-
short	0_2	0_2
int, float	00_2	00_2
double	00_2	000_2
long double	00_2	000_2
char*	00_2	000_2

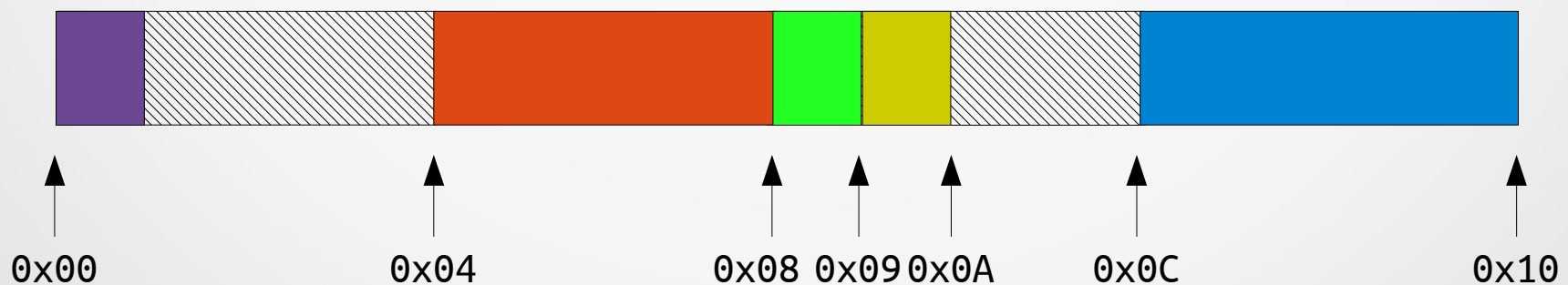
Data Alignment

- Highly probable exam question.
 - Given a struct, calculate the number of bytes it will consume:

```
struct someStruct{  
    char oneChar;  
    int someInt;  
    char someChar;  
    char someOtherChar;  
    char* aPtr;  
}
```

Data Alignment

```
struct someStruct{  
    char oneChar;  
    int someInt;  
    char someChar;  
    char someOtherChar;  
    char* aPtr;  
}
```



Size : 16 bytes

Data Alignment

- Highly probable exam **followup** question.
 - Given a struct, minimize the number of bytes it occupies

```
struct someStruct{  
    char oneChar;  
    int someInt;  
    char someChar;  
    char someOtherChar;  
    char* aPtr;  
}
```

Data Alignment

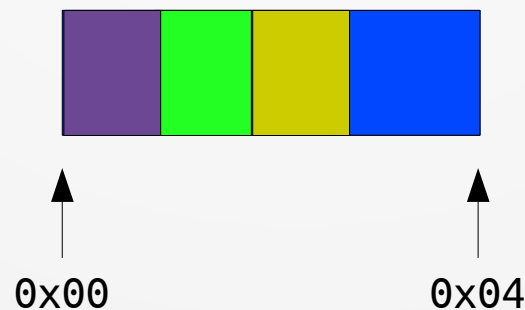
```
struct someStruct{  
    int someInt;  
    char* aPtr;  
    char oneChar;  
    char someChar;  
    char someOtherChar;  
}
```



Size : Only 12 bytes

Unions

```
union someStruct{  
    int someInt;  
    char* aPtr;  
    char oneChar;  
    char someChar;  
    char someOtherChar;  
}
```



Size : 4 bytes