

Final Exam Review

CIT 593

What to expect?

1.5 Hour Exam, Closed book and notes

Anything specific needed will be provided

- LC3 ISA instructions
- ASCII table

Concentrate on material after midterm

- Similar format seen on quizzes and midterm
- But LC3 ISA instructions and coding is still expected
- Questions related to C and LC3
 - C to LC3 and vice versa
 - Activation record of function

CIT593

2

Compilation Process

Entire mechanism is usually called the **compiler**

Preprocessor

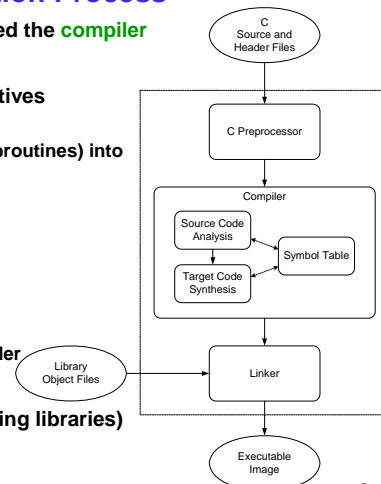
- Acts upon C preprocessor directives
 - Eg. #include or #define
 - Inserts contents of file (e.g. subroutines) into the current source file
- “Source-level” transformations
 - Output is still C

Compiler

- Generates object file
 - Machine instructions (binaries)
 - Similar processor LC3 Assembler

Linker

- Combine object files (including libraries) into executable image



CIT593

3

Differences between C and Java

C Language

- No Boolean type
- Pointers, and Pointer Arithmetic
- No Safety Checking
- #include instead of import
- Limited Object Orientation
 - Can have data structure with fields but no member functions
 - No way to make data private or protected

CIT593

4

Variable Properties

Identifier: variable name

Type: how data is interpreted, and how much space it needs

Scope: is the region of the program in which the variable is alive and accessible

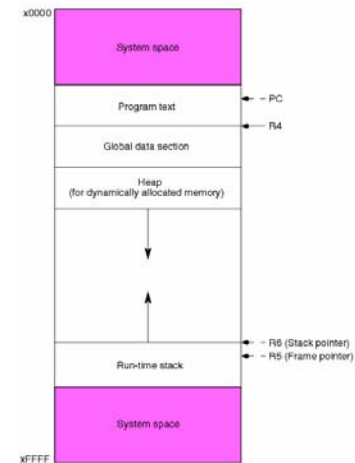
Storage: how C compiler allocates storage and whether or not the variable loses its value when the block that contains it has completed execution

CIT593

5

Memory Allocation in C for Variables

- Global Data section for global variables (static class storage)
- Run-time stack section for local variables (automatic class storage)
- Heap section for dynamically allocated memory (memory allocated at runtime)



CIT593

6

Operators

Three things to know about each operator

(1) Function

- What does it do?

(2) Precedence

- In which order are operators combined?
- Example:
"a * b + c * d" is the same as "(a * b) + (c * d)"
because multiply (*) has a higher precedence than addition (+)

(3) Associativity

- In which order are operators of the same precedence combined?
- Example:
"a - b - c" is the same as "(a - b) - c"
because add/sub associate left-to-right

CIT593

7

Logical Operators

Symbol	Operation	Usage	Precedence	Assoc
!	logical NOT	!x	4	r-to-l
&&	logical AND	x && y	14	l-to-r
	logical OR	x y	15	l-to-r

Treats entire variable (or value) as

- TRUE (non-zero), or
- FALSE (zero)

Result is 1 (TRUE) or 0 (FALSE)

- `x = 15; y = 0; printf("%d", x || y);`

Bit-wise vs Logical

- `1 & 8 = 0` (000001 AND 001000 = 000000)
- `1 && 8 = 1` (True & True = True)

CIT593

8

Evaluation Order special case in C

What does this do?

```
void func(int x, int y) { printf("%d %d", x, y); }

int main()
{
    int x = 3;
    func(x, x++);
    ...
}
```

Answer: **undefined!**

- Displays either "3 3" or "4 3"
- Why? C does not define order of evaluation in such a case
- Depends on compiler writer
- Suggestion: **don't** modify variable in parameter list

CIT593

9

Correspondence between Ptr and Array Notation

```
int data[5]
int * cptr = data;
```

each line below gives three equivalent expressions:

<u>cptr</u>	<u>data</u>	<u>&data[0]</u>
(cptr + 3)	(data + 3)	&data[3]
*cptr	*data	data[0]
*(cptr + 3)	*(data + 3)	data[3]

CIT593

10

String Declaration

String is array of characters terminated by null character

What's the difference between:

- char amessage[] = "message"
- char *pmessage = "message"

Answer:

```
m e s s a g e \0
```

- char amessage[] = "message" // single array

```
□ → m e s s a g e \0
```

- char *pmessage = "message" // pointer and array

CIT593

11

String Copy - Array Style

```
void strcpy(char dest[], char src[])
{
    int i = 0;
    while ((dest[i] = src[i]) != '\0') {
        i++;
    }
}
```

CIT593

12

String Copy - Pointer Style

```
void strcpy(char* dest, char* src)
{
    while ((*dest = *src) != '\\0') {
        dest++;
        src++;
    }
}
```

CIT593

13

<string.h>

Know the following functions in string.h:

- `int strlen(char* str)`
- `void strcpy(char* dest, char* src)`
- `int strcmp(char* s1, char* s2)`
 - Returns 0 on equal, -1 or 1 if greater or less
 - Remember, 0 is false, so equal returns false!

CIT593

14

Beware

Array are not the same as pointers although they may look like

what is the difference between arrays and pointers?

- Arrays automatically allocate space, but can't be relocated or resized.
- Pointers must be explicitly assigned to point to allocated space but can be reassigned (i.e. pointed at different objects) at will, and have many other uses besides serving as the base of blocks of memory.

CIT593

15

Main(), revisited

Main supports command line parameters

In Java

```
public static void main(String[] args)
```

Example:

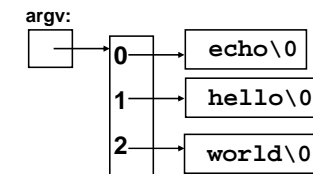
```
gcc -o echo.c echo
./echo hello world
```

In C:

```
int main(int argc, char *argv[])
{
}

```

An array of strings



By convention argv[0] is name by which program is invoked so argc is atleast 1.

CIT593

16

Declaring a struct

We name the struct and declare “fields” (or “members”)

```
struct w_type {  
    int highTemp;  
    int lowTemp;  
    double precip;  
    double windSpeed;  
    int windDirection;  
};
```

This is declaration so no memory is actually allocated yet!

CIT593

17

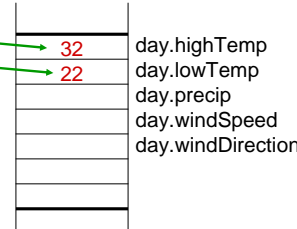
Defining and Using a struct

We define a variable using our new data type as follows:

```
struct w_type day;
```

Memory is now allocated (on stack), and we can access individual fields of this variable

```
day.highTemp = 32;  
day.lowTemp = 22;
```



Struct declaration allocates a contiguous memory for the collection

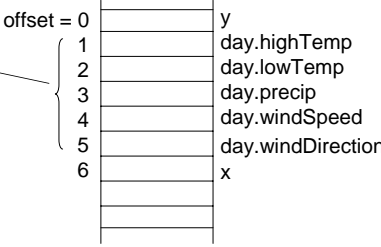
CIT593

18

Memory allocation for struct on Runtime Stack

Consider the following code

```
...  
int x;  
WeatherData day;  
int y;  
  
day.highTemp = 12;  
day.lowTemp = 1;  
day.windDirection = 3;  
...  
offset = 0  
1  
2  
3  
4  
5  
6  
y  
day.highTemp  
day.lowTemp  
day.precip  
day.windSpeed  
day.windDirection  
x
```



Note: $\text{addr}(x) = R5 + \text{offset} = R5 + 6$

R5 = Frame Pointer

CIT593

19

Dynamic Allocation

Problem

- What if we don't know the number of days for our weather program?
- Can't allocate array, because don't know maximum number of days that might be required
- Even if we do know the maximum number, it might be wasteful to allocate that much memory because most of the time only a few days' worth of data is needed

Solution

- Allocate storage dynamically, as needed

CIT593

20

Problems w/ explicit storage deallocation

There are two potential errors when deallocating (freeing) storage yourself (as in C):

- Deallocating too soon, so that you have *dangling references* (pointers to storage that has been freed and possibly reused for something else)
- Forgetting to deallocate, so that unused storage accumulates and you have no more heap memory left during runtime of your program