SE 292 (3:0)
High Performance Computing
Aug. 2014

R. Govindarajan (govind@serc)
Yogesh Simmhan (simmhan@serc)

Syllabus

Introduction to computer systems: Processors, Memory, I/O Devices; Cost, timing and scale (size) models; Program Execution: Machine level view of a program; typical RISC instruction set and execution; Pipelining. Performance issues and Techniques; Caching and Virtual Memory. Temporal and spatial locality. Typical compiler optimizations. Identifying program bottlenecks – profiling, tracing. Simple high level language optimizations – locality enhancement, memory disambiguation. Choosing appropriate computing platforms: benchmarking, cost-performance issues. [8 - weeks]

OS Concepts: Process Management, System Calls, Memory Management, and I/O. *Parallel Computing*: Introduction to parallel architectures and interconnection networks, communication latencies. Program parallelization; task partitioning and mapping; data distribution; message passing; synchronization and deadlocks. Distributed memory programming using MPI/PVM. Shared memory parallel programming. Multithreading. [8 weeks]

Course Objectives

- Understand computer systems (processor architecture and memory subsystem (incl. cache) from a programmer's viewpoint to maximize performance of his/her appln.
- Understand underlying Operating Systems concepts (programmer's viewpoint) – process, memory management, file system, …
- Understand parallel processing systems (programmer's viewpoint) and Parallel Prog.

$\Rightarrow$ Improve Application Performance

Reference Material

- Bryant and O'Hallaron, Computer Systems: A Programmer's Perspective. Pearson Education (2003)
- Dowd, High Performance Computing, O'Reilly (1993)
- Selected readings from other sources
  - Silberschatz, Galvin and Gagne, Operating System Concepts (8th Edition), John Wiley & Sons.
  - Ananth Grama, Anshul Gupta, George Karypis, Vipin Kumar, Introduction to Parallel Computing. Addison Wesley, 2003.
  - David Culler, Jaswant Singh, Parallel Computer Architecture. Morgan Kauffman

Course Conduct

- Lectures Tuesday, Thursday
- Official 8.00 a.m. – 9.30 a.m.

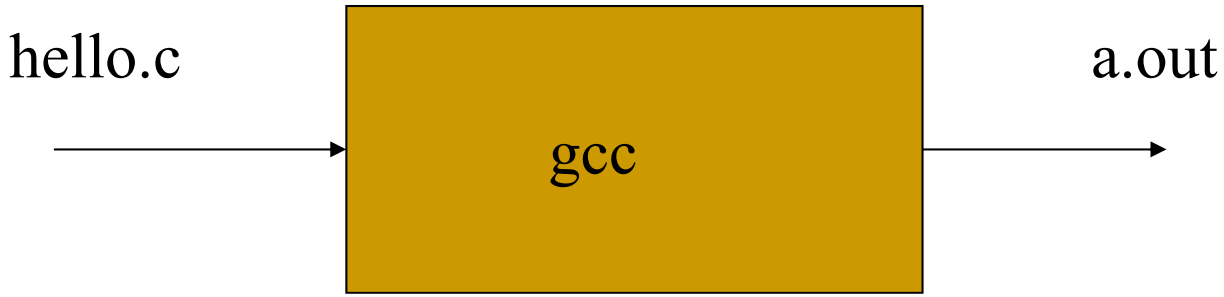
- Assignments & Projects : 25%
- Midterms : 11/9, 14/10, 11/11 30%
- Final Exam : Dec ?? 45%

What is a Computer Program?

- Description of algorithms and data structures
- Programming language: standard notation for writing programs
- Examples: C, Java, assembly language, machine language
- Need for program translators

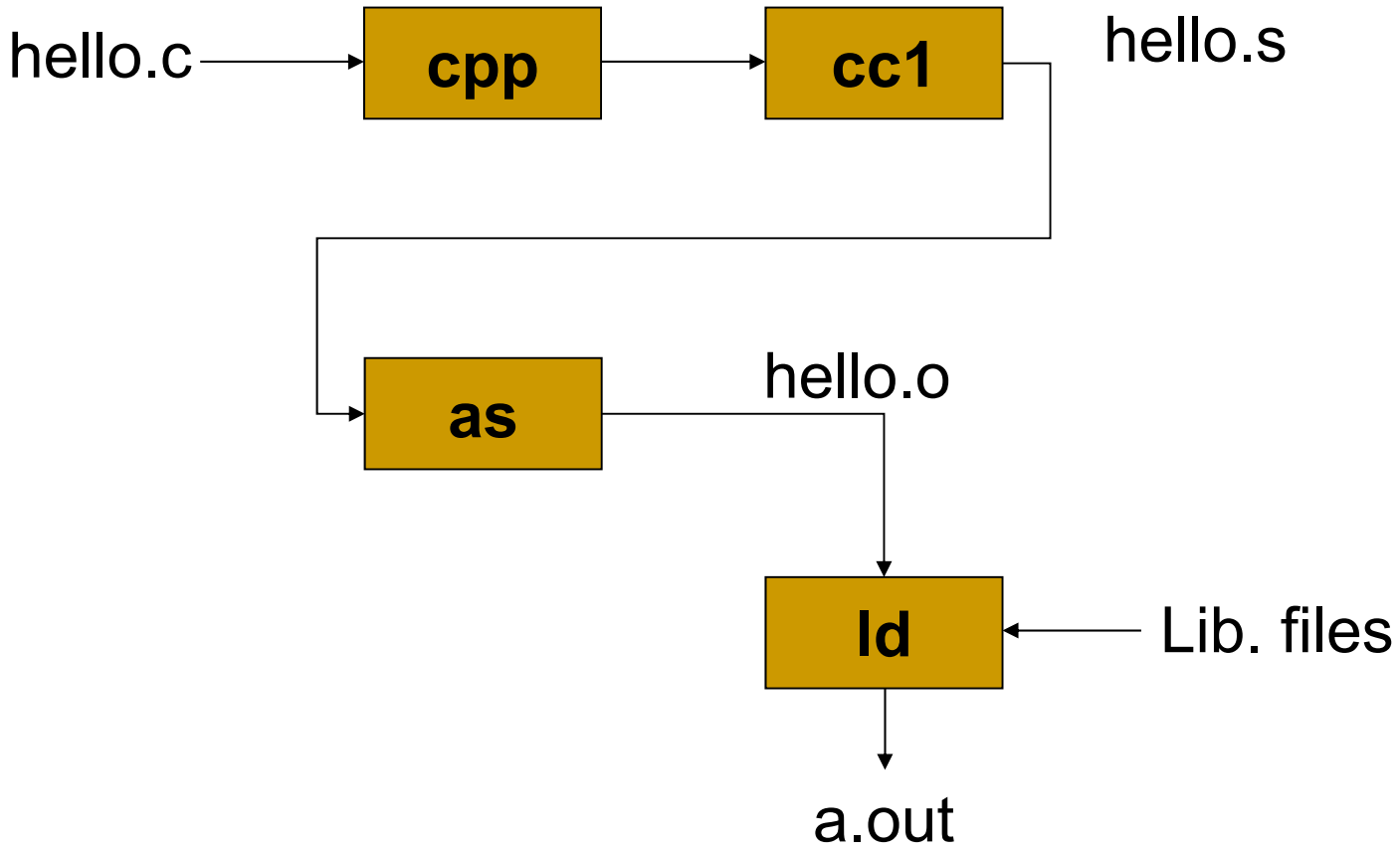
What are the Steps in gcc?

% gcc hello.c



`source file'** **`object file'

Steps in gcc and Intermediate Files



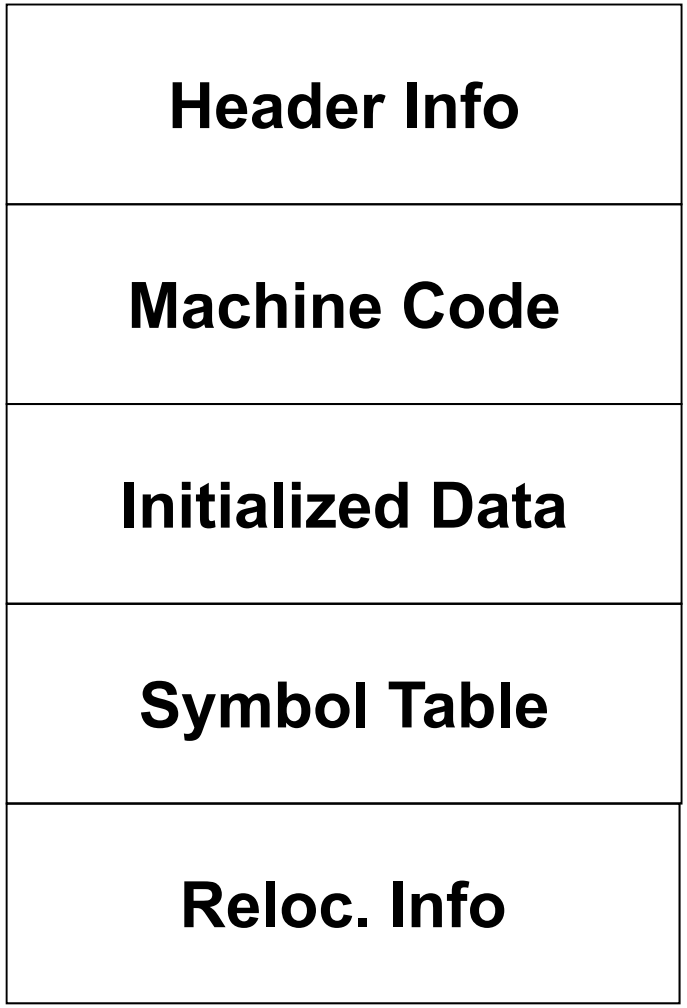
Steps in gcc

- `cpp, cc1, as, ld`
- Temporary files generated and used
- `a.out` well defined format
- Contents?
  - Program (machine instructions)
  - Data values (values, size of arrays)
  - Other info needed for
    - execution
    - relocation
    - debugging

Example

```
#include<stdio.h>
#include<math.h>
float arr[100];
int size=100;
void main()
{
        int i;
        float sum;
        for(i=0, sum=0.0; i<size; i++)
        {
                arr[i]  =  sqrt(arr[i]);
                sum +=  arr[i];
        }
        printf ("sum is %f\n",sum);
}
```

Object file format



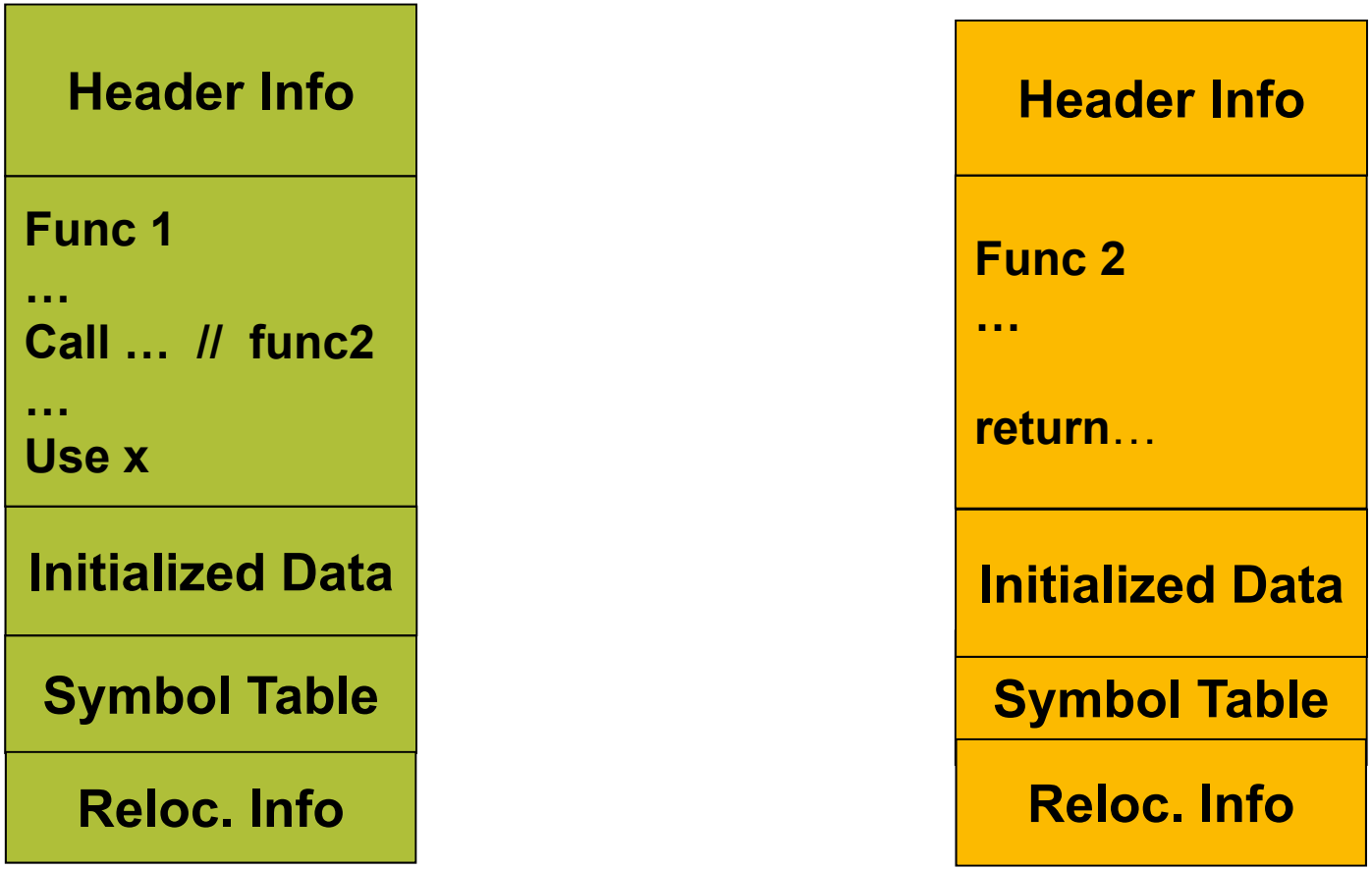
Format of the Object File

Header Info.	0	94	MachineCode Size
	4	4	Init. Data size
	8	400	Uninit. Data size
	12	60	Symbol Table
	16	??	Reloc. Info
Machine Code	20	??	Start of main; Code for First intrn.
	…	…	
	66	??	code for call sqrt

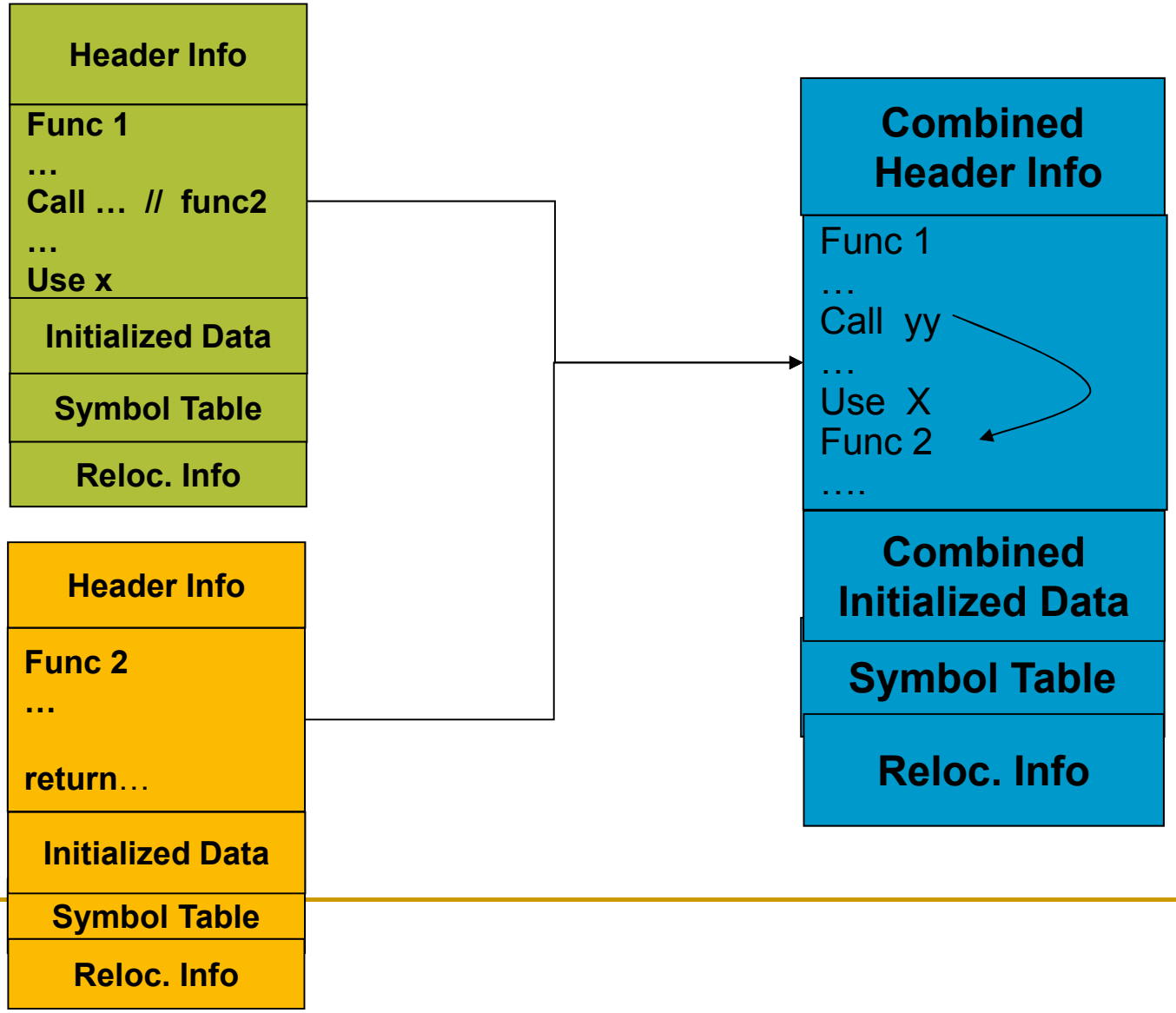
Format of the Object File

Init. Data	114	100	--	Initialized Data
Symbol Table	118	XX	size	Name of symbol “size” and its addr..
	130	YY	arr	Name of symbol “arr” and its addr..
	142	ZZ	main	Name of symbol “main” and its addr..
	154	??	Sqrt	Name of symbol “sqrt” and its addr..
	166	??	printf	Name of symbol “printf” and its addr..
Reloc. Info	178	??	Info. On offsets at which ext. vars are called	

Linking Multiple Modules

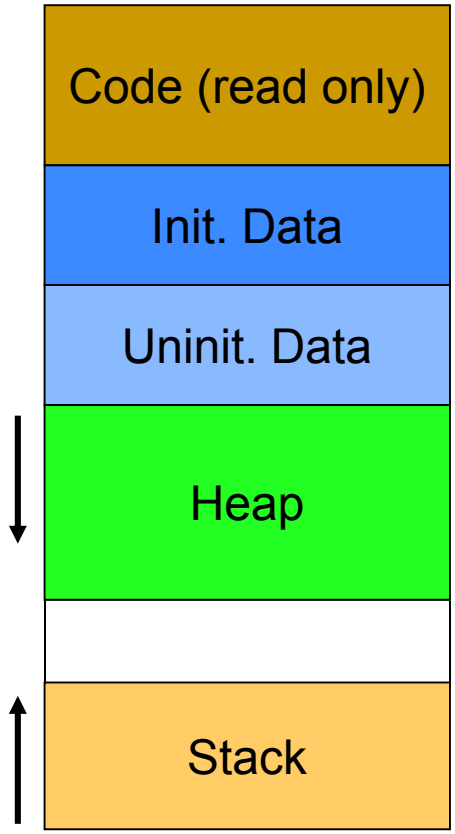


Linking Multiple Modules



Process' Address Space

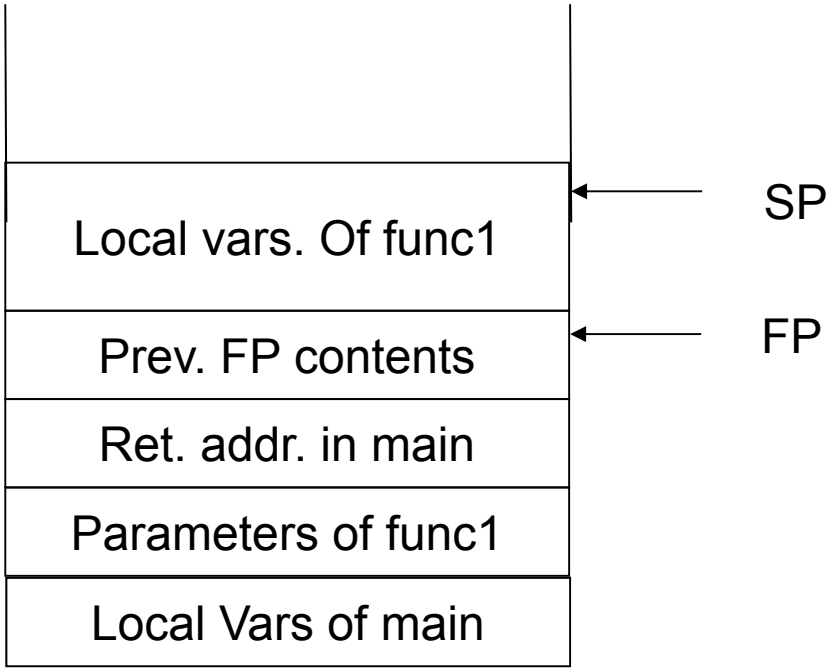
- Process: A program in execution, containing
  - code
  - data (initialized and uninitialized)
  - heap (for dynamically allocated data)
  - stack (for local variables, function parameters, …)



Function Call and Return

- Transfer of control from point of function call in the **caller** to start of called function (**callee**)
- Execution of function body until point of return
- Transfer of control back to just after the point of call in the caller
- Parameter passing: by value, by reference, etc
- Formal parameters and actual parameters
- Function local variables

Function Call and Return



Homework

- Find out details of a.out format in your machine
- What is the size of a.out from hello.c?
- How is the old SP value recovered on a function return?
- Where are different types of variables (automatic, static, global, dynamic variables) stored?

Program Data: Different Kinds

- Constant vs variable
- Basic vs structured
- Of different types
  - Character
  - Integer (unsigned, signed)
  - Real
  - Others (boolean, complex, …)

Of Different Lifetimes

- Execution time of program
  - Initialized/uninitialized data
  - Must be indicated in executable file
- Explicit creation until deletion
  - Dynamic memory allocation
  - malloc, free
  - Why and where?
- During execution of a function
- Memory is Static/Heap/Stack allocated

How is Data Represented?

- Binary, Bit (b), Byte (B), K, M, G, T
- Character data representation: ASCII code
  - 7 bit code, with an added parity bit
  - (HW: Write a C program to generate the ASCII code)

Integer Data

- Signed vs Unsigned integer
- Representing a signed integer
  - Sign-magnitude representation

$$X_{n-1} X_{n-2} \cdots X_2 X_1 X_o$$

least significant bit

represents the value

$$(-1)^{x_{n-1}} \times \sum_{i=0}^{n-2} x_i 2^i$$

2s Complement Representation

The *n* bit quantity

$$x_{n-1}x_{n-2}\cdots x_0$$

represents the signed integer value

$$-x_{n-1}2^{n-1}+\sum_{i=0}^{n-2}x_i 2^i$$

Hex Rep.	Bin. Rep.	Dec. Val.
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	-8
9	1001	-7
A	1010	-6
B	1011	-5
C	1100	-4
D	1101	-3
E	1110	-2
F	1111	-1

Which Representation is Better?

- Considerations
  - speed of arithmetic (addition, multiplication)
  - comparison
  - range of values that can be represented
- 2s complement is widely used

Example: Signed integer

What is the Decimal value of 0xED7E

1110 1101 0111 1110

Invert all bits and Add 1

0001 0010 1000 0001

1

0001 0010 1000 0010

which is 4738 decimal

Answer: -4738 decimal

What is the 2s compl. Rep. for -4738

4738 is 0x1282

0001 0010 1000 0010

Invert all bits and Add 1

1110 1101 0111 1101

1

1110 1101 0111 1110

0xED7E

which is the rep. -4738

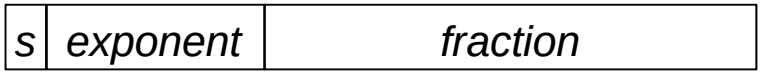
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Real Data: Floating Point Representation

IEEE Floating Point Standard (IEEE 754)

32 bit value with 3 components

- *s* (1 bit sign)
- *e* (8 bit exponent) – excess 127 notation;
  range -126 to 127
- *f* (23 bit fraction)



represents the value

$$(-1)^s \times 1.f \times 2^{e-127}$$

An Example

Consider the value 0.625

- Equal to 0.101 in binary 1.01×2^{-1}
- s: 0, e: 126, f: 010…000
- In 32 bits,
  0 01111110 01000000000000000000000

More on IEEE Floating Point

- Why is the exponent represented in this way? (excess-127 representation for signed integers)
- Normalized representation
- Special forms
  - Denormalized values (exp = 0; f = non-zero)
  - Zero (exp = 0; f = 0)
  - Infinity (exp = 255; f = 0)
  - NaN (exp = 255; f = non-zero)

More on IEEE Floating Point

- Smallest positive number (in normalized form)

  $(1+0) * 2^{-126} = 1.2 * 10^{-38}$

- Largest positive number (in normalized form)

  $(1+1) * 2^{127} = 3.4 * 10^{38}$

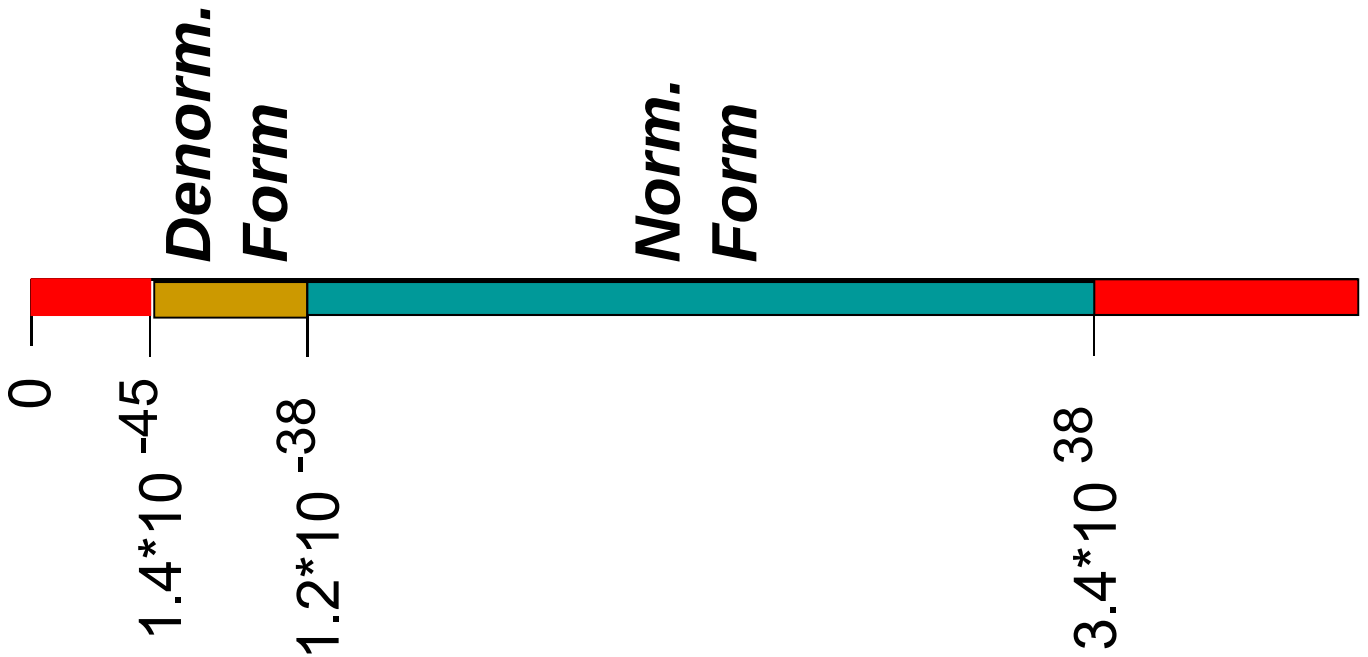
- Smallest postive number in denormalized representation

  $2^{-23} * 2^{-126} = 1.4 * 10^{-45}$

  (Implicit exponent of 2^{-126} for denorm values)

  - Denorm form for representing numbers from $1.4 * 10^{-45}$ to $1.2 * 10^{-38}$

Floating Point Representation: Summary



What about Double Precision FP?

IEEE Double Precision Floating Point Representation (IEEE 754)

64 bit value with 3 components

- *s* (1 bit sign)
- *e* (11 bit exponent) – excess 1023 notation;
  range -1022 to 1023
- *f* (52 bit fraction)

represents the value

$$(-1)^s \times 1.f \times 2^{e-1023}$$

Homework

1. Read B&O Chapter 2
2. B&O 2.43
3. B&O 2.46
4. B&O 2.49
5. B&O 2.53
6. Write a C program which checks whether the machine it is running on supports IEEE FP representation and arithmetic on NaN, denormals and infinity.
7. Write a C program to find the machine epsilon.

Assignments (so far)

- Find out details of a.out format for any C program.
- Write a C program to identify in which region the following types of variables are stored:

  (a) global (b) local; (c) static, and (d) dynamically allocated generate the ASCII code)
- Write a C program which checks whether the machine it is running on supports IEEE FP representation and arithmetic on NaN, denormals and infinity
- Write a C program to find the machine epsilon.