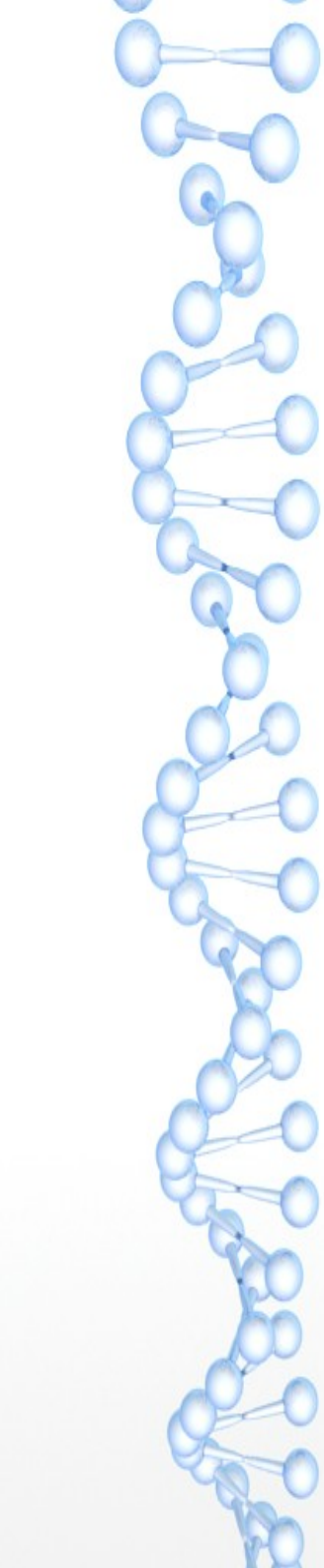


Principles and Techniques of Compilers
FINAL PROJECT

Siqing Hou

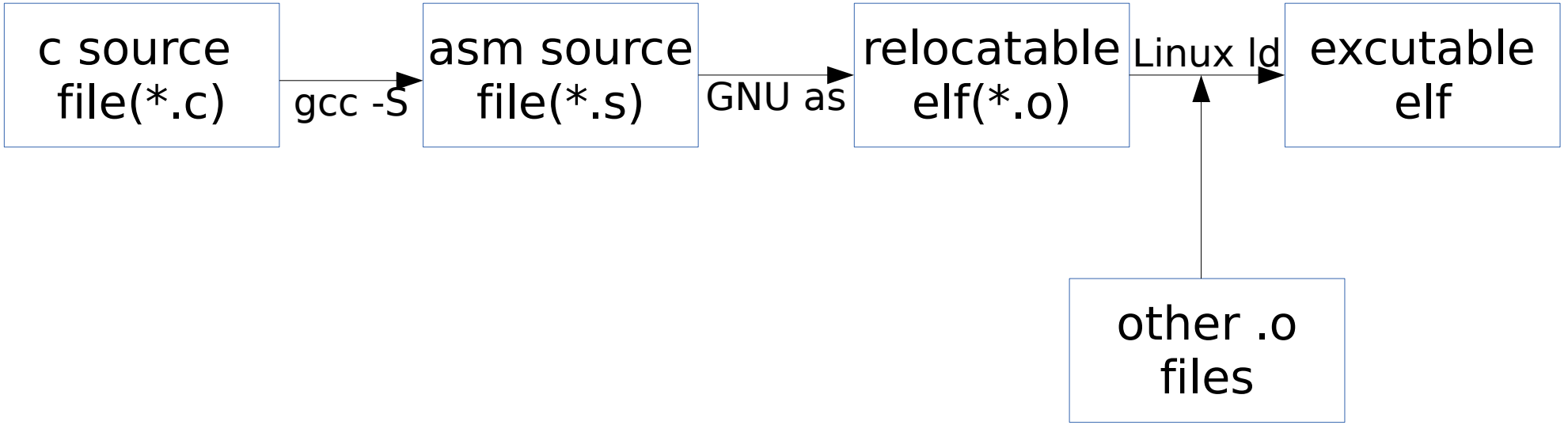
1. C-linkable i386 assembly code generator

2. OpenMP auto parallel using GNU OpenMP

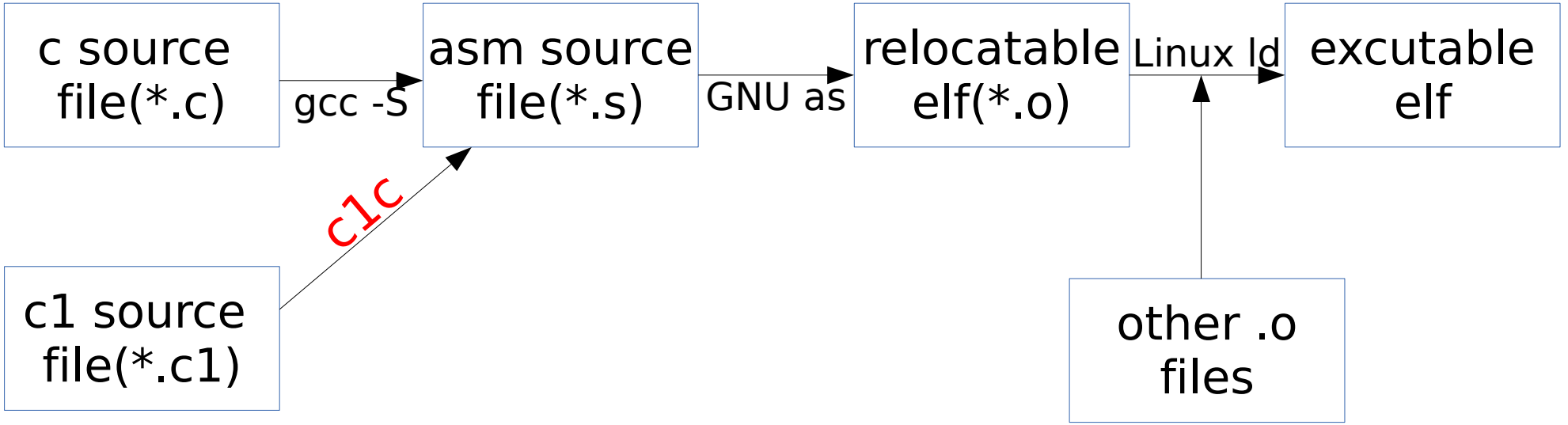
Tools Used

- Flex
- Bison
- GNU C compiler &GNU assembler
- libgomp GNU OpenMP Library

x86 Code Generator



x86 Code Generator



Instructions Used

- Labels:
  - Global var/const label
  - Function name label
  - .LFB .LFE Function begin/end label

Instructions Used

- Pseudo Instructions:
  - .file
  - .comm (bss allocate)
  - .text/.data/.section .rodata
  - .globl .size .align .long .zero .type
- No .cfi pseudo instructions

Instructions Used

- leal movl pushl popl
- addl subl imull idivl negl testl cmpl cltd
- jmp je jne jge jle jg jl jp jnp
- call leave ret
- rep stosl

Static Memory Allocation

- Const
  - Use rodata section
  - .long .zero
- Var
  - Use bss section
  - .comm

Stack Memory Allocation

- Calculate size when compiling, allocate only once for each function
- Space optimizing: use max depth instead of total size
- Initialize local const when executing

Extern function

- C source file

```
void f(){
   …
}
```

- C1 source file

```
extern void f();
void main()
   {
   f()
   }
```

```
        .file   "allocate.c1"
        .comm   a,4,4
        .comm   b,4,4
        .comm   c,40,16
        .globl  d
        .section        .rodata
        .align  4
        .type   d, @object
        .size   d, 4
d:
        .long   1
        .globl  e
        .section        .rodata
        .align  4
        .type   e, @object
        .size   e, 12
e:
        .long   1
        .long   2
        .long   5
        .globl  f
        .section        .rodata
        .align  64
        .type   f, @object
        .size   f, 64
f:
```

```
sub:
.LFB0:
        pushl   %ebp
        movl    %esp, %ebp
        subl    $0, %esp
        movl    $333, c+12
        addl    $0, %esp
        leave
        ret
.LFE0:
        .size   sub, .-sub
        .text
        .globl  testfunc
        .type   testfunc, @function
testfunc:
.LFB1:
        pushl   %ebp
        movl    %esp, %ebp
        subl    $3392, %esp
        pushl   %edi
        leal    -808(%ebp), %edx
        movl    $0, %eax
        movl    $100, %ecx
        movl    %edx, %edi
        rep stosl
        movl    $1, -808(%ebp)
        popl    %edi
        pushl   %edi
        leal    -848(%ebp), %edx
```

C1 QuickSort

```
speedrace of c & c1 quicksort
c1 version

real    0m6.178s
user    0m6.136s
sys     0m0.036s
c version

real    0m4.509s
user    0m4.440s
sys     0m0.064s
```

C1 stdio library

```c
#include <stdio.h>
extern int data;

void output(){
        printf("%d",data);
}
void input(){
        scanf("%d",&data);
}

void space(){
        printf(" ");
}

void nextline(){
        printf("\n");
}
```

OpenMP Parallel

- Fork-Join thread model
  - parallel clause
  - for clause
  - parallel clause

GOMP_parallel

- void GOMP_parallel(

  (void *)subfunction(void *data),

  (void *)data,

  int,

  int

  );

Statment ->Function

- Compile parallel stmt/block to a subfunction
- Pass %ebp to subfuction
  - Get hostfunction's %ebp to %esi
  - Use %esi to locate hostfunction's variables
  - Use %ebp to locate private variables

OpenMP runtime libraries

- omp_get_num_threads
- omp_get_thread_num

OpenMP Hello

```
hello world from thread 0
hello world from thread 2
hello world from thread 3
hello world from thread 1
```

Parallel Computing

```
thread id 0 calculate sum from 0 to 249999
thread id 3 calculate sum from 750000 to 999999
thread id 2 calculate sum from 500000 to 749999
thread id 1 calculate sum from 250000 to 499999
thread id 0 :thread sum is 765951
thread id 2 :thread sum is 767707
thread id 3 :thread sum is 758906
thread id 1 :thread sum is 757022
sum = 3049586
time used:98240 usec
```