Table of Contents

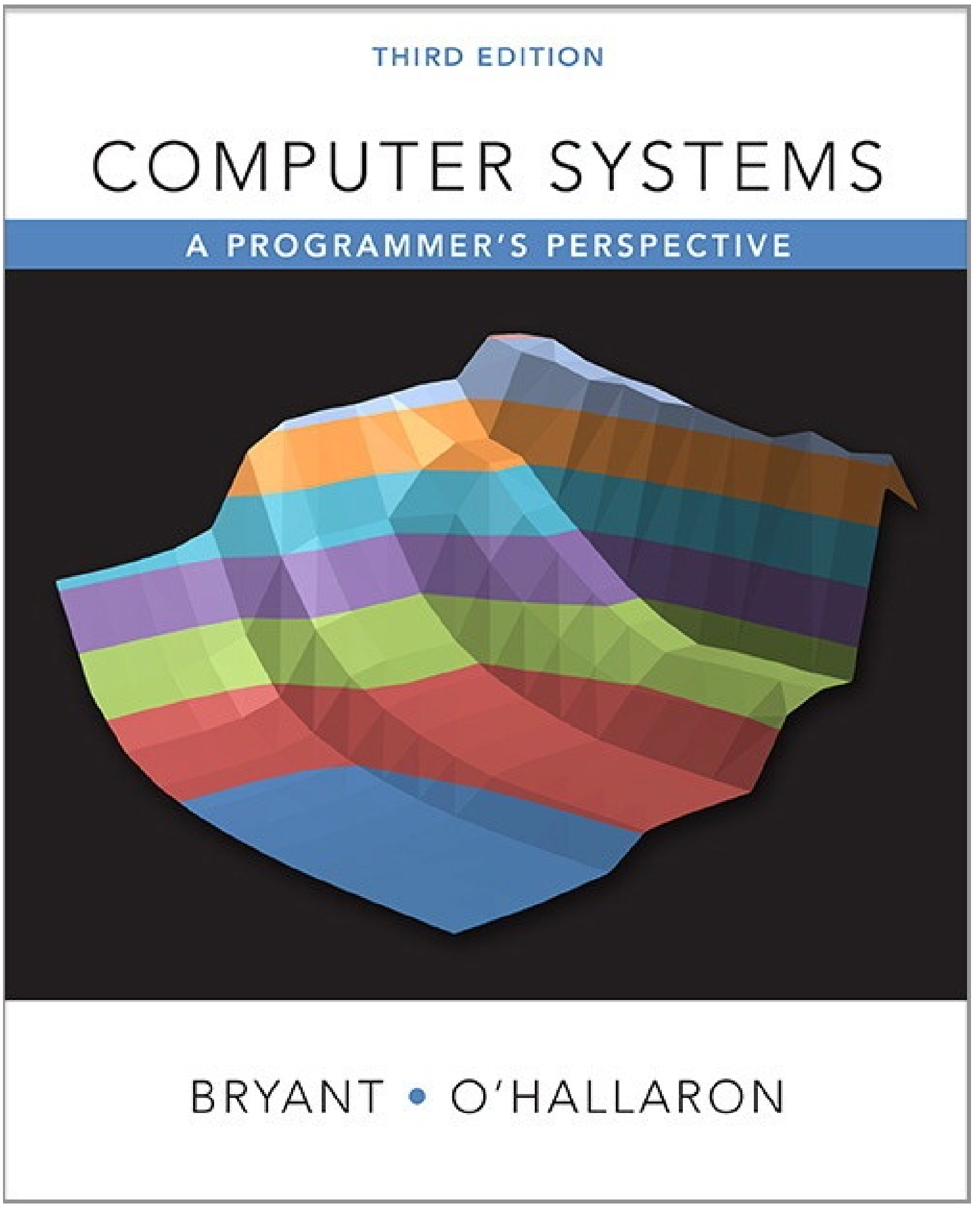
Part 1 Program Structure and Execution

Part 2 Running Programs on a System

Part 3 Interaction and Communication between Programs

CSAPP-3e-Solutions build passing

Computer Systems: A Programmer's Perspective Third Edition Solutions



at first

Almost all solutions has its own code piece in c/gas/yas and every code piece is tested!

Code files are classified by chapter. Please visit the index page of every chapter to see more info.

issues

> Hurry makes faulty work and no improvement makes it disappointed.

Thanks every improvement makes the project better.

issue	from	by	status
disqus 2.63 comment	https://dreamanddead.gitbooks.io/csapp-3e-solutions/chapter2/2.63.html	hhdx	discussing

build

prerequisite

- x64 linux system
- docker

pull env image

```
sudo docker pull dreamanddead/csapp-3e-solutions
```

code

clone code

```
git clone https://github.com/DreamAndDead/CSAPP-3e-Solutions.git
cd CSAPP-3e-Solutions
```

compile

```
make
```

test

```
make test
```

clean

```
make clean
```

gitbook

must install gitbook plugins first before other gitbook actions

```
make plugin
```

serve book in http://localhost:4000

```
make serve
```

generate site in ./_book/

```
make html
```

generate E-books in ./

```
make pdf
make mobi
make epub
```

feedback

If you encounter some problem, you can email me or comment on disqus in specific solution page

license

GPLv3

at last

I'll be :) if this little book helps you and make your life more convenient.

Long Live Open Source.

A Tour of Computer Systems

> Computer science is no more about computers than astronomy is about telescopes.
>
> by Edsger Dijkstra

no homework here.

Representing and Manipulating Information

> Everything is physics and math.
>
> by Katherine Johnson

2.1 - 2.54 visit book

2.55 - 2.97 visit here

code directory: `./code`

test way:

- assert means assert function from `<assert.h>`
- output means to watch code output to judge if it works right

solution	code file	test way
2.55	show-bytes.c	output
2.56	show-bytes.c	output
2.57	show-bytes-more.c	output
2.58	is-little-endian.c	assert
2.59	generate-a-word.c	assert
2.60	replace-byte.c	assert
2.61	2.61.c	assert
2.62	int-shifts-are-arithemetic.c	assert
2.63	srl-sra.c	assert
2.64	any-odd-one.c	assert
2.65	odd-ones.c	assert
2.66	leftmost-one.c	assert
2.67	int-size-is-32.c	assert
2.68	lower-one-mask.c	assert

2.69	rotate-left.c	assert
2.70	fits-bits.c	assert
2.71	xbyte.c	assert
2.72	copy-int.c	assert
2.73	saturating-add.c	assert
2.74	tsub-ok.c	assert
2.75	unsigned-high-prod.c	assert
2.76	calloc.c	assert
2.77	2.77.c	assert
2.78	divide-power2.c	assert
2.79	mul3div4.c	assert
2.80	threeforths.c	assert
2.81	2.81.c	assert
2.82	2.81.c	assert
2.83	-----	-----
2.84	float-le.c	assert
2.85	-----	-----
2.86	-----	-----
2.87	-----	-----
2.88	-----	-----
2.89	2.89.c	assert
2.90	fpwr2.c	assert
2.91	-----	-----
2.92	floats/float-negate.c	assert
2.93	floats/float-absval.c	assert
2.94	floats/float-twice.c	assert
2.95	floats/float-half.c	assert
2.96	floats/float-f2i.c	assert
2.97	floats/float-i2f.c	assert

2.55

```
/*
 * show-bytes.c
 */

#include <stdio.h>

typedef unsigned char* byte_pointer;

void show_bytes(byte_pointer start, size_t len) {
  size_t i;
  for (i = 0; i < len; i++) {
    printf(" %.2x", start[i]);
  }
  printf("\n");
}

void show_int(int x) {
  show_bytes((byte_pointer) &x, sizeof(int));
}

void show_float(float x) {
  show_bytes((byte_pointer) &x, sizeof(float));
}

void show_pointer(void* x) {
  show_bytes((byte_pointer) &x, sizeof(void*));
}

void test_show_bytes(int val) {
  int ival = val;
  float fval = (float) ival;
  int* pval = &ival;

  show_int(ival);
  show_float(fval);
  show_pointer(pval);
```

```
}

int main(int argc, char* argv[]) {
  int test_num = 328;

  test_show_bytes(test_num);
  return 0;
}
```

uname -mr:

```
4.4.26-gentoo x86_64
```

compile:

```
gcc -m64 show-bytes.c -o show-bytes
```

run:

```
./show-bytes
```

output:

```
 48 01 00 00
 00 00 a4 43
 a8 1e 71 ee fc 7f 00 00
```

2.56

change

```
int test_num = 1024;
```

run:

```
./show-bytes
```

output:

```
00 04 00 00
00 00 80 44
c8 fe 83 2f fc 7f 00 00
```

try more integers :)

2.57

```c
/*
 * show-bytes.c
 */

#include <stdio.h>

typedef unsigned char* byte_pointer;

void show_bytes(byte_pointer start, size_t len) {
  size_t i;
  for (i = 0; i < len; i++) {
    printf(" %.2x", start[i]);
  }
  printf("\n");
}

void show_int(int x) {
  show_bytes((byte_pointer) &x, sizeof(int));
}

void show_float(float x) {
  show_bytes((byte_pointer) &x, sizeof(float));
}

void show_pointer(void* x) {
  show_bytes((byte_pointer) &x, sizeof(void*));
}

//=============
// 2.57 changes
//=============
void show_short(short x) {
  show_bytes((byte_pointer) &x, sizeof(short));
}

void show_long(long x) {
  show_bytes((byte_pointer) &x, sizeof(long));
}
```

```
void show_double(double x) {
  show_bytes((byte_pointer) &x, sizeof(double));
}
//==================
// 2.57 changes end
//==================

void test_show_bytes(int val) {
  int ival = val;
  float fval = (float) ival;
  int* pval = &ival;

  show_int(ival);
  show_float(fval);
  show_pointer(pval);

  //==============
  // 2.57 changes
  //==============
  short sval = (short) ival;
  long lval = (long) ival;
  double dval = (double) ival;

  show_short(sval);
  show_long(lval);
  show_double(dval);
  //==================
  // 2.57 changes end
  //==================
}

int main(int argc, char* argv[]) {
  int test_num = 328;

  test_show_bytes(test_num);
  return 0;
}
```

uname -rm

```
4.4.0-21-generic x86_64
```

run

```
./show-bytes-more
```

output

```
48 01 00 00
00 00 a4 43
18 b7 2e 20 fd 7f 00 00
48 01
48 01 00 00 00 00 00 00
00 00 00 00 00 80 74 40
```

2.58

```c
/*
 * is-little-endian.c
 */

#include <stdio.h>
#include <assert.h>

typedef unsigned char* byte_pointer;

int is_little_endian() {
  int test_num = 0xff;
  byte_pointer byte_start = (byte_pointer) &test_num;

  if (byte_start[0] == 0xff) {
    return 1;
  }
  return 0;
}

int main(int argc, char* argv[]) {
  assert(is_little_endian());
  return 0;
}
```

2.59

expression

```
(x & 0xFF) | (y & ~0xFF)
```

try it

```
/*
 * generate-a-word.c
 */

#include <stdio.h>
#include <assert.h>

int main(int argc, char* argv[]) {
  size_t mask = 0xff;
  size_t x = 0x89ABCDEF;
  size_t y = 0x76543210;

  size_t res = (x & mask) | (y & ~mask);
  assert(res == 0x765432EF);

  return 0;
}
```

2.60

```c
/*
 * replace-byte.c
 */

#include <stdio.h>
#include <assert.h>

unsigned replace_byte(unsigned x, int i, unsigned char b) {
  if (i < 0) {
    printf("error: i is negetive\n");
    return x;
  }
  if (i > sizeof(unsigned)-1) {
    printf("error: too big i");
    return x;
  }

  // 1 byte has 8 bits, << 3 means * 8
  unsigned mask = ((unsigned) 0xFF) << (i << 3);
  unsigned pos_byte = ((unsigned) b) << (i << 3);

  return (x & ~mask) | pos_byte;
}

int main(int argc, char *argv[]) {
  unsigned rep_0 = replace_byte(0x12345678, 0, 0xAB);
  unsigned rep_3 = replace_byte(0x12345678, 3, 0xAB);

  assert(rep_0 == 0x123456AB);
  assert(rep_3 == 0xAB345678);
  return 0;
}
```

2.61

A

```
!~x
```

B

```
!x
```

C

```
!~(x | ~0xff)
```

D

```
!((x >> ((sizeof(int)-1) << 3)) & 0xff)
```

test it

```c
/*
 * 2.61.c
 */

#include <stdio.h>
#include <assert.h>

int A(int x) {
  return !~x;
}

int B(int x) {
  return !x;
}

int C(int x) {
  return A(x | ~0xff);
```

```
}

int D(int x) {
  return B((x >> ((sizeof(int)-1) << 3)) & 0xff);
}

int main(int argc, char* argv[]) {
  int all_bit_one = ~0;
  int all_bit_zero = 0;

  assert(A(all_bit_one));
  assert(!B(all_bit_one));
  assert(C(all_bit_one));
  assert(!D(all_bit_one));

  assert(!A(all_bit_zero));
  assert(B(all_bit_zero));
  assert(!C(all_bit_zero));
  assert(D(all_bit_zero));

  // test magic number 0x1234ff
  assert(!A(0x1234ff));
  assert(!B(0x1234ff));
  assert(C(0x1234ff));
  assert(D(0x1234ff));

  // test magic number 0x1234
  assert(!A(0x1234));
  assert(!B(0x1234));
  assert(!C(0x1234));
  assert(D(0x1234));

  return 0;
}
```

2.62

```
/*
 * int-shifts-are-arithemetic.c
 */

#include <stdio.h>
#include <assert.h>

int int_shifts_are_arithemetic() {
  int num = -1;
  return !(num ^ (num >> 1));
}

int main(int argc, char* argv[]) {
  assert(int_shifts_are_arithemetic());
  return 0;
}
```

2.63

```c
/*
 * srl-sra.c
 */

#include <stdio.h>
#include <assert.h>

unsigned srl(unsigned x, int k) {
  unsigned xsra = (int) x >> k;

  int w = sizeof(int) << 3;
  int mask = (int) -1 << (w - k);
  return xsra & ~mask;
}

int sra(int x, int k) {
  int xsrl = (unsigned) x >> k;

  int w = sizeof(int) << 3;
  int mask = (int) -1 << (w - k);
  if (x < 0) {
    return xsrl | mask;
  }
  return xsrl;
}

int main(int argc, char* argv[]) {
  unsigned test_unsigned = 0x12345678;
  int test_int = 0x12345678;

  assert(srl(test_unsigned, 4) == test_unsigned >> 4);
  assert(sra(test_int, 4) == test_int >> 4);

  return 0;
}
```

2.64

```c
/*
 * any-odd-one.c
 */
#include <stdio.h>
#include <assert.h>

int any_odd_one(unsigned x) {
  return !!(0xAAAAAAAA & x);
}

int main(int argc, char* argv[]) {
  assert(any_odd_one(0x2));
  assert(!any_odd_one(0x4));
  return 0;
}
```

2.65

thanks this anwser on stackoverflow

```
/*
 * odd-ones.c
 */
#include <stdio.h>
#include <assert.h>

int odd_ones(unsigned x) {
  x ^= x >> 16;
  x ^= x >> 8;
  x ^= x >> 4;
  x ^= x >> 2;
  x ^= x >> 1;
  x &= 0x1;
  return x;
}

int main(int argc, char* argv[]) {
  assert(odd_ones(0x10101011));
  assert(!odd_ones(0x01010101));
  return 0;
}
```

2.66

```c
/*
 * leftmost-one.c
 */
#include <stdio.h>
#include <assert.h>

/*
 * Generate mask indicating leftmost 1 in x. Assume w=32
 * For example, 0xFF00 -> 0x8000, and 0x6000 -> 0x4000.
 * If x = 0, then return 0
 */
int leftmost_one(unsigned x) {
  /*
   * first, generate a mask that all bits after leftmost one are
 one
   * e.g. 0xFF00 -> 0xFFFF, and 0x6000 -> 0x7FFF
   * If x = 0, get 0
   */
  x |= x >> 1;
  x |= x >> 2;
  x |= x >> 4;
  x |= x >> 8;
  x |= x >> 16;
  /*
   * then, do  mask & (~mask >> 1), reserve leftmost bit one
   * that's we want
   */
  return x & (~x >> 1);
}

int main(int argc, char* argv[]) {
  assert(leftmost_one(0xFF00) == 0x8000);
  assert(leftmost_one(0x6000) == 0x4000);
  return 0;
}
```

2.67

A.

In section 6.5.7 Bitwise shift operators of c11 standard, it said

> If the value of the right operand is negative or is greater than or equal to the width of the promoted left operand, the behavior it undefined.

B.

see function int_size_is_32

C.

see function int_size_is_32_for_16bit

```c
/*
 * int-size-is-32.c
 */

#include <stdio.h>
#include <assert.h>

/* The following function does not run properly on some machine
*/
/*
int bad_int_size_is_32() {
  int set_msb = 1 << 31;
  int beyond_msb = 1 << 32;

  return set_msb && !beyond_msb;
}
*/

int int_size_is_32() {
  int set_msb = 1 << 31;
  int beyond_msb = set_msb << 1;

  return set_msb && !beyond_msb;
}

int int_size_is_32_for_16bit() {
  int set_msb = 1 << 15 << 15 << 1;
  int beyond_msb = set_msb << 1;

  return set_msb && !beyond_msb;
}

int main(int argc, char *argv[]) {
  assert(int_size_is_32());
  assert(int_size_is_32_for_16bit());
  return 0;
}
```

2.68

```c
/*
 * lower-one-mask.c
 */
#include <stdio.h>
#include <assert.h>

/*
 * Mask with least signficant n bits set to 1
 * Example: n = 6 -> 0x3F, n = 17 -> 0x1FFFF
 * Assume 1 <= n <= w
 */
int lower_one_mask(int n) {
  int w = sizeof(int) << 3;
  return (unsigned) -1 >> (w - n);
}

int main(int argc, char* argv[]) {
  assert(lower_one_mask(6) == 0x3F);
  assert(lower_one_mask(17) == 0x1FFFF);
  assert(lower_one_mask(32) == 0xFFFFFFFF);
  return 0;
}
```

2.69.md

```c
/*
 * rotate-left.c
 */
#include <stdio.h>
#include <assert.h>

/*
 * Do rotate left shift. Assume 0 <= n < w
 * Example when x = 0x12345678 and w = 32:
 *    n = 4 -> 0x23456781, n = 20 -> 0x67812345
 */
unsigned rotate_left(unsigned x, int n) {
  int w = sizeof(unsigned) << 3;
  /* pay attention when n == 0 */
  return x << n | x >> (w - n - 1) >> 1;
}

int main(int argc, char* argv[]) {
  assert(rotate_left(0x12345678, 4) == 0x23456781);
  assert(rotate_left(0x12345678, 20) == 0x67812345);
  return 0;
}
```

2.70

```
/*
 * fit-bits.c
 */
#include <stdio.h>
#include <assert.h>

int fits_bits(int x, int n) {
  /*
   * Assume w = 8, n = 3
   * if x > 0
   *   0b00000110 is ok, 0b00001010 is not
   *   first w-n bits must be 0
   * if x < 0
   *   0b11111100 is ok, 0b10111100 is not, and 0b11111000 is no
t yet
   *   first w-n+1 bits must be 1
   */
  int w = sizeof(int) << 3;
  x >>= n - 1;
  /*
   * !(x >> 1) return 1 only when all bits are 0,
   * !~x == 1 only when all bits are 1
   */
  return !(x >> 1) || !~x;
}

int main(int argc, char* argv[]) {
  assert(fits_bits(0xFF, 8));
  assert(!fits_bits(0xFFFFFF00, 8));
  return 0;
}
```

2.71

A.

This function can't extract a negetive byte number from word

B.

```
/*
 * xbyte.c
 */
#include <stdio.h>
#include <assert.h>

typedef unsigned packet_t;

int xbyte(packet_t word, int bytenum) {
  /*
   * pay attention when byte we want is negetive
   *
   * Assume sizeof(unsigned) is 4
   * first shift left 8 * (4 - 1 - bytenum)
   * then arthemetic shift right 8 * (4 - 1) reserve signficant
bit
   */
  int size = sizeof(unsigned);
  int shift_left_val = (size - 1 - bytenum) << 3;
  int shift_right_val = (size - 1) << 3;
  return (int) word << shift_left_val >> shift_right_val;
}

int main(int argc, char* argv[]) {
  assert(xbyte(0xAABBCCDD, 1) == 0xFFFFFFCC);
  assert(xbyte(0x00112233, 2) == 0x11);
  return 0;
}
```

2.72

A.

sizeof(val) return type size_t, it usually is a kind of unsigned type.

calculate maxbytes-sizeof(val) get size_t type value, and it always >= 0

B.

```c
/*
 * copy-int.c
 */
#include <stdio.h>
#include <assert.h>
#include <string.h>
#include <stdlib.h>

void copy_int(int val, void* buf, int maxbytes) {
  /* compare two signed number, avoid someone set maxbytes a neg
etive value */
  if (maxbytes >= (int) sizeof(val)) {
    memcpy(buf, (void*)&val, sizeof(val));
  }
}

int main(int argc, char* argv[]) {
  int maxbytes = sizeof(int) * 10;
  void* buf = malloc(maxbytes);
  int val;

  val = 0x12345678;
  copy_int(val, buf, maxbytes);
  assert(*(int*)buf == val);

  val = 0xAABBCCDD;
  copy_int(val, buf, 0);
  assert(*(int*)buf != val);

  free(buf);
  return 0;
}
```

2.73

I can't figure out a elegant solution :(

thanks https://zhangjunphy.github.io/csapp/chap2.html :)

```
/*
 * saturating-add.c
 */
#include <stdio.h>
#include <assert.h>
#include <limits.h>

int saturating_add(int x, int y) {
  int sum = x + y;
  int sig_mask = INT_MIN;
  /*
   * if x > 0, y > 0 but sum < 0, it's a positive overflow
   * if x < 0, y < 0 but sum >= 0, it's a negetive overflow
   */
  int pos_over = !(x & sig_mask) && !(y & sig_mask) && (sum & si
g_mask);
  int neg_over = (x & sig_mask) && (y & sig_mask) && !(sum & sig
_mask);

  (pos_over && (sum = INT_MAX) || neg_over && (sum = INT_MIN));

  return sum;
}

int main(int argc, char* argv[]) {
  assert(INT_MAX == saturating_add(INT_MAX, 0x1234));
  assert(INT_MIN == saturating_add(INT_MIN, -0x1234));
  assert(0x11 + 0x22 == saturating_add(0x11, 0x22));
  return 0;
}
```

2.74

```
/*
 * tsub-ok.c
 */
#include <stdio.h>
#include <assert.h>
#include <limits.h>

/* Determine whether arguments can be substracted without overfl
ow */
int tsub_ok(int x, int y) {
  if (y == INT_MIN) {
    return 0;
  }

  int neg_y = -y;
  int sum = x + neg_y;
  int pos_over = x > 0 && neg_y > 0 && sum < 0;
  int neg_over = x < 0 && neg_y < 0 && sum >= 0;

  return !(pos_over || neg_over);
}

int main(int argc, char* argv[]) {
  assert(!tsub_ok(0x00, INT_MIN));
  assert(tsub_ok(0x00, 0x00));
  return 0;
}
```

2.75.md

```
/*
 * unsigned-high-prod.c
 */
#include <stdio.h>
#include <assert.h>
#include <inttypes.h>

int signed_high_prod(int x, int y) {
  int64_t mul = (int64_t) x * y;
  return mul >> 32;
}

unsigned unsigned_high_prod(unsigned x, unsigned y) {
  /* TODO calculations */
  int sig_x = x >> 31;
  int sig_y = y >> 31;
  int signed_prod = signed_high_prod(x, y);
  return signed_prod + x * sig_y + y * sig_x;
}

/* a theorically correct version to test unsigned_high_prod func
 */
unsigned another_unsigned_high_prod(unsigned x, unsigned y) {
  uint64_t mul = (uint64_t) x * y;
  return mul >> 32;
}

int main(int argc, char* argv[]) {
  unsigned x = 0x12345678;
  unsigned y = 0xFFFFFFFF;

  assert(another_unsigned_high_prod(x, y) == unsigned_high_prod(
x, y));
  return 0;
}
```

2.76.md

```
/*
 * calloc.c
 */
#include <stdio.h>
#include <assert.h>
#include <stdlib.h>
#include <stdint.h>
#include <string.h>

/* rename to avoid conflict */
void* another_calloc(size_t nmemb, size_t size) {
  if (nmemb == 0 || size == 0) {
    return NULL;
  }
  size_t buf_size = nmemb * size;
  /* a good way to check overflow or not */
  if (nmemb == buf_size / size) {
    void* ptr = malloc(buf_size);
    memset(ptr, 0, buf_size);
    return ptr;
  }
  return NULL;
}

int main(int argc, char* argv[]) {
  void* p;
  p = another_calloc(0x1234, 1);
  assert(p != NULL);
  free(p);

  p = another_calloc(SIZE_MAX, 2);
  assert(p == NULL);
  free(p);
  return 0;
}
```

2.77

```
/*
 * 2.77.c
 */
#include <stdio.h>
#include <assert.h>

/* K = 17 */
int A(int x) {
  return (x << 4) + x;
}

/* K = -7 */
int B(int x) {
  return x - (x << 3);
}

/* K = 60 */
int C(int x) {
  return (x << 6) - (x << 2);
}

/* K = -112 */
int D(int x) {
  return (x << 4) - (x << 7);
}

int main(int argc, char* argv[]) {
  int x = 0x87654321;
  assert(A(x) == 17 * x);
  assert(B(x) == -7 * x);
  assert(C(x) == 60 * x);
  assert(D(x) == -112 * x);
  return 0;
}
```

2.78

```c
/*
 * divide-power2.c
 */
#include <stdio.h>
#include <assert.h>
#include <limits.h>

/*
 * Divide by power of 2, -> x/2^k
 * Assume 0 <= k < w-1
 */
int divide_power2(int x, int k) {
  int is_neg = x & INT_MIN;
  (is_neg && (x = x + (1 << k) - 1));
  return x >> k;
}

int main(int argc, char* argv[]) {
  int x = 0x80000007;
  assert(divide_power2(x, 1) == x / 2);
  assert(divide_power2(x, 2) == x / 4);
  return 0;
}
```

2.79

```c
/*
 * mul3div4.c
 */
#include <stdio.h>
#include <assert.h>
#include <limits.h>

/*
 * code from 2.78
 *
 * Divide by power of 2, -> x/2^k
 * Assume 0 <= k < w-1
 */
int divide_power2(int x, int k) {
  int is_neg = x & INT_MIN;
  (is_neg && (x = x + (1 << k) - 1));
  return x >> k;
}

int mul3div4(int x) {
  int mul3 = (x << 1) + x;
  return divide_power2(mul3, 2);
}

int main(int argc, char* argv[]) {
  int x = 0x87654321;
  assert(mul3div4(x) == x * 3 / 4);
  return 0;
}
```

2.80

```
/*
 * threeforths.c
 */
#include <stdio.h>
#include <assert.h>
#include <limits.h>

/*
 * calculate 3/4x, no overflow, round to zero
 *
 * no overflow means divide 4 first, then multiple 3, diffrent f
rom 2.79 here
 *
 * rounding to zero is a little complicated.
 * every int x, equals f(first 30 bit number) plus l(last 2 bit
number)
 *
 *   f = x & ~0x3
 *   l = x & 0x3
 *   x = f + l
 *   threeforths(x) = f/4*3 + l*3/4
 *
 * f doesn't care about round at all, we just care about roundin
g from l*3/4
 *
 *   lm3 = (l << 1) + l
 *
 * when x > 0, rounding to zero is easy
 *
 *   lm3d4 = lm3 >> 2
 *
 * when x < 0, rounding to zero acts like divide_power2 in 2.78
 *
 *   bias = 0x3    // (1 << 2) - 1
 *   lm3d4 = (lm3 + bias) >> 2
 */
int threeforths(int x) {
  int is_neg = x & INT_MIN;
```

```
  int f = x & ~0x3;
  int l = x & 0x3;

  int fd4 = f >> 2;
  int fd4m3 = (fd4 << 1) + fd4;

  int lm3 = (l << 1) + l;
  int bias = (1 << 2) - 1;
  (is_neg && (lm3 += bias));
  int lm3d4 = lm3 >> 2;

  return fd4m3 + lm3d4;
}

int main(int argc, char* argv[]) {
  assert(threeforths(8) == 6);
  assert(threeforths(9) == 6);
  assert(threeforths(10) == 7);
  assert(threeforths(11) == 8);
  assert(threeforths(12) == 9);

  assert(threeforths(-8) == -6);
  assert(threeforths(-9) == -6);
  assert(threeforths(-10) == -7);
  assert(threeforths(-11) == -8);
  assert(threeforths(-12) == -9);
  return 0;
}
```

2.81

A.

```
-1 << k
```

B.

```
~(-1 << k) << j
```

```c
/*
 * 2.81.c
 */
#include <stdio.h>
#include <assert.h>

/* Assume 0 <= k < w */
int A(int k) {
  return -1 << k;
}

/* Assume 0 <= j,k < w */
int B(int k, int j) {
  return ~A(k) << j;
}

int main(int argc, char* argv[]) {
  assert(A(8) == 0xFFFFFF00);
  assert(B(16, 8) == 0x00FFFF00);
  return 0;
}
```

2.82

A.

wrong, when x is INT_MIN

B.

right

C.

right

D.

right

E.

right

```c
/*
 * 2.82.c
 */
#include <stdio.h>
#include <assert.h>
#include <limits.h>
#include "lib/random.h"

/* broken when x is INT_MIN */
int A(int x, int y) {
  return (x < y) == (-x > -y);
}

/*
 * right
 *
 * ((x + y) << 4) + y - x
 *   =>
 * x << 4 - x + y << 4 + y
 *   =>
 * x*16 - x + y*16 + y
```

```
 *   whether overflow or not, =>
 * x*15 + y*17
 */
int B(int x, int y) {
  return ((x + y) << 4) + y - x == 17 * y + 15 * x;
}

/*
 * right
 *
 * ~x + ~y + 1
 *   =>
 * ~x + 1 + ~y + 1 - 1
 *   =>
 * -x + -y - 1
 *   =>
 * -(x + y) - 1
 *   =>
 * ~(x + y) + 1 - 1
 *   =>
 * ~(x + y)
 */
int C(int x, int y) {
  return ~x + ~y + 1 == ~(x + y);
}

/*
 * right
 *
 * (ux - uy) == -(unsigned) (y - x)
 *   =>
 * -(ux - uy) == (unsigned) (y - x)
 *   =>
 * (ux - uy) == (unsigned) (x - y)
 */
int D(int x, int y) {
  unsigned ux = (unsigned) x;
  unsigned uy = (unsigned) y;

  return (ux - uy) == -(unsigned) (y - x);
```

```
}

/*
 * right
 *
 * x >> 2 << 2
 *   =>
 * x & ~0x3
 *   =>
 * x - num(00/01/10/11)
 *   =>
 * ((x >> 2) << 2) <= x
 */
int E(int x, int y) {
  return ((x >> 2) << 2) <= x;
}

int main(int argc, char* argv[]) {
  init_seed();
  int x = random_int();
  int y = random_int();

  assert(!A(INT_MIN, 0));
  assert(B(x, y));
  assert(C(x, y));
  assert(D(x, y));
  assert(E(x, y));
  return 0;
}
```

2.83

A.

```
n = 0.yyyyy...

n << k = y.yyyyy... = Y + n

n << k - n = Y

n = Y/(2^k - 1)
```

B.

(a).

y = 101, Y = 5, k = 3

n = 5/7

(b).

y = 0110, Y = 6, k = 4

n = 2/5

(c).

y = 010011, Y = 19, k = 6

n = 19/63

2.84

thanks czy1996

```
/*
 * float-le.c
 */
#include <stdio.h>
#include <assert.h>

unsigned f2u(float x) {
  return *(unsigned*)&x;
}

int float_le(float x, float y) {
  unsigned ux = f2u(x);
  unsigned uy = f2u(y);

  unsigned sx = ux >> 31;
  unsigned sy = uy >> 31;

  // ref: https://github.com/DreamAndDead/CSAPP-3e-Solutions/iss
ues/1
  return (ux << 1 == 0 && uy << 1 == 0) || /* both zeros */
    (sx && !sy) ||                         /* x < 0, y >= 0 or x
 <= 0, y > 0 */
    (!sx && !sy && ux <= uy) ||            /* x > 0, y >= 0 or x
 >= 0, y > 0 */
    (sx && sy && ux >= uy);                /* x < 0, y <= 0 or x
 <= 0, y < 0 */
}

int main(int argc, char* argv[]) {
  assert(float_le(-0, +0));
  assert(float_le(+0, -0));
  assert(float_le(0, 3));
  assert(float_le(-4, -0));
  assert(float_le(-4, 4));
  return 0;
}
```

2.85

bias = 2^(k-1) - 1

V = 2^E * M

A.

7.0 = 0b111.000...

M = 0b1.11, f = 0b0.11, E = 2, e = bias + E, V = 7.0

bits

```
0 10....01 110....
```

B.

Assume bias >> n

bigest odd number, M must be 0b1.111111...., f = 0b0.11111111...(n bits 1)

E = n, V = 0b11111111...(n+1 bits 1) = 2^(n+1) - 1

bits

```
0 bias+n 11111....
```

C.

least standard number

M must be 0b1.00...., f = 0b0.000...., E = 1 - bias

V = 2^(1-bias)

reciprocal

V = 2^(bias-1)

M = 0b1.0000..., f = 0b0.000...., E = bias-1, e = bias + E

bits

```
0 11...101 00000.....
```

2.86

bias = 2^(15-1) - 1

description	binary	decimal
least positive unstandard	0 0000...(15) 0 000...(62)1	2^(1-bias-63)
least positive standard	0 000...(14)1 1 000....(63)	2^(1-bias)
bigest standard	0 111...(14)0 1 111...(63)	2^bias * (2-2^-63)

2.87

Desc	Hex	M	E	V	D
-0	0x8000	0	-14	-0	-0.0
>2 least	0x4001	1025/1024	1	1025/512	2.00195312
512	0x6000	1	9	512	512.0
bigest denormalized	0x03FF	1023/1024	-14	1023/(2^24)	6.09755516e-5
-oo	0xFC00	-	-	-oo	-oo
ox3BB0	0x3BB0	123/64	-1	123/128	0.9609375

2.88

A bit	A value	B bit	B value
1 01110 001	-9/16	1 0110 0010	-9/16
0 10110 101	13*2^4	0 1110 1010	13*2^4
1 00111 110	-7/2^10	1 0000 0111	-7/2^10
0 00000 101	5/2^11	0 0000 0001	1/2^10
1 11011 000	-2^12	1 1110 1111	-31*2^3
0 11000 100	3*2^8	0 1111 0000	+oo

2.89

```
/*
 * 2.89.c
 */
#include <stdio.h>
#include <assert.h>
#include <limits.h>
#include "lib/random.h"

/*
 * most important thing is that all double number come from ints
 */

/* right */
int A(int x, double dx) {
  return (float)x == (float)dx;
}

/* wrong when y is INT_MIN */
int B(int x, double dx, int y, double dy) {
  return dx-dy == (double)(x-y);
}

/* right */
int C(double dx, double dy, double dz) {
  return (dx+dy)+dz == dx+(dy+dz);
}

/*
 * wrong
 *
 * FIXME I don't know what conditions cause false
 */
int D(double dx, double dy, double dz) {
  return (dx*dy)*dz == dx*(dy*dz);
}

/* wrong when dx != 0 and dz == 0 */
int E(double dx, double dz) {
```

```c
  return dx/dx == dz/dz;
}

int main(int argc, char* argv[]) {
  init_seed();

  int x = random_int();
  int y = random_int();
  int z = random_int();
  double dx = (double)x;
  double dy = (double)y;
  double dz = (double)z;

  printf("%x %x %x\n", x, y, z);

  assert(A(x, dx));
  assert(!B(0, (double)(int)0, INT_MIN, (double)(int)INT_MIN));
  assert(C(dx, dy, dz));
  /* magic number, brute force attack */
  assert(!D((double)(int)0x64e73387, (double)(int)0xd31cb264, (double)(int)0xd22f1fcd));
  assert(!E(dx, (double)(int)0));
  return 0;
}
```

2.90

```
/*
 * fpwr2.c
 */
#include <stdio.h>
#include <assert.h>
#include <math.h>

float u2f(unsigned x) {
  return *(float*) &x;
}

/* 2^x */
float fpwr2(int x) {
  /* Result exponent and fraction */
  unsigned exp, frac;
  unsigned u;

  if (x < 2-pow(2,7)-23) {
    /* too small. return 0.0 */
    exp = 0;
    frac = 0;
  } else if (x < 2-pow(2,7)) {
    /* Denormalized result */
    exp = 0;
    frac = 1 << (unsigned)(x - (2-pow(2,7)-23));
  } else if (x < pow(2,7)-1+1) {
    /* Normalized result */
    exp = pow(2,7)-1+x;
    frac = 0;
  } else {
    /* Too big, return +oo */
    exp = 0xFF;
    frac = 0;
  }

  /* pack exp and frac into 32 bits */
  u = exp << 23 | frac;
  /* Result as float */
```

```
  return u2f(u);
}

int main(int argc, char* argv[]) {
  assert(fpwr2(0) == powf(2,0));
  assert(fpwr2(100) == powf(2,100));
  assert(fpwr2(-100) == powf(2,-100));
  assert(fpwr2(10000) == powf(2,10000));
  assert(fpwr2(-10000) == powf(2,-10000));
  return 0;
}
```

2.91

A.

```
0x40490FDB

0 10000000 10010010000111111011011
```

float number

```
0b11.0010010000111111011011
```

B.

ref 2.83

```
0b11.001001(001)...
```

C.

9th

2.92

```c
/*
 * float-negate.c
 */
#include <stdio.h>
#include <assert.h>
#include "float-negate.h"

float_bits float_negate(float_bits f) {
  unsigned sig = f >> 31;
  unsigned exp = f >> 23 & 0xFF;
  unsigned frac = f & 0x7FFFFF;

  int is_NAN = (exp == 0xFF) && (frac != 0);
  if (is_NAN) {
    return f;
  }

  return ~sig << 31 | exp << 23 | frac;
}
```

2.93

```
/*
 * float-absval.c
 */
#include <stdio.h>
#include <assert.h>
#include "float-absval.h"

float_bits float_absval(float_bits f) {
  unsigned sig = f >> 31;
  unsigned exp = f >> 23 & 0xFF;
  unsigned frac = f & 0x7FFFFF;

  int is_NAN = (exp == 0xFF) && (frac != 0);
  if (is_NAN) {
    return f;
  }

  return 0 << 31 | exp << 23 | frac;
}
```

2.94

```c
/*
 * float-twice.c
 */
#include <stdio.h>
#include <assert.h>
#include "float-twice.h"

float_bits float_twice(float_bits f) {
  unsigned sig = f >> 31;
  unsigned exp = f >> 23 & 0xFF;
  unsigned frac = f & 0x7FFFFF;

  int is_NAN_or_oo = (exp == 0xFF);
  if (is_NAN_or_oo) {
    return f;
  }

  if (exp == 0) {
    /* Denormalized */
    frac <<= 1;
  } else if (exp == 0xFF - 1) {
    /* twice to oo */
    exp = 0xFF;
    frac = 0;
  } else {
    /* Normalized */
    exp += 1;
  }

  return sig << 31 | exp << 23 | frac;
}
```

2.95

```
/*
 * float-half.c
 */
#include <stdio.h>
#include <assert.h>
#include "float-half.h"

float_bits float_half(float_bits f) {
  unsigned sig = f >> 31;
  unsigned rest = f & 0x7FFFFFFF;
  unsigned exp = f >> 23 & 0xFF;
  unsigned frac = f & 0x7FFFFF;

  int is_NAN_or_oo = (exp == 0xFF);
  if (is_NAN_or_oo) {
    return f;
  }

  /*
   * round to even, we care about last 2 bits of frac
   *
   * 00 => 0 just >>1
   * 01 => 0 (round to even) just >>1
   * 10 => 1 just >>1
   * 11 => 1 + 1 (round to even) just >>1 and plus 1
   */
  int addition = (frac & 0x3) == 0x3;

  if (exp == 0) {
    /* Denormalized */
    frac >>= 1;
    frac += addition;
  } else if (exp == 1) {
    /* Normalized to denormalized */
    rest >>= 1;
    rest += addition;
    exp = rest >> 23 & 0xFF;
    frac = rest & 0x7FFFFF;
```

```
  } else {
    /* Normalized */
    exp -= 1;
  }

  return sig << 31 | exp << 23 | frac;
}
```

2.96

this is exactly how my machine does!!!

```
/*
 * float-f2i.c
 */
#include <stdio.h>
#include <assert.h>
#include "float-f2i.h"

/*
 * Compute (float) f
 * If conversion cause overflow or f is NaN, return 0x80000000
 */
int float_f2i(float_bits f) {
  unsigned sig = f >> 31;
  unsigned exp = f >> 23 & 0xFF;
  unsigned frac = f & 0x7FFFFF;
  unsigned bias = 0x7F;

  int num;
  unsigned E;
  unsigned M;

  /*
   * consider positive numbers
   *
   * 0 00000000 00000000000000000000000
   *   ===>
   * 0 01111111 00000000000000000000000
   *   0 <= f < 1
   * get integer 0
   *
   * 0 01111111 00000000000000000000000
   *   ===>
   * 0 (01111111+31) 00000000000000000000000
   *   1 <= f < 2^31
   * integer round to 0
   *
```

```
   * 0 (01111111+31) 00000000000000000000000
   *   ===>
   * greater
   *   2^31 <= f < oo
   * return 0x80000000
   */
  if (exp >= 0 && exp < 0 + bias) {
    /* number less than 1 */
    num = 0;
  } else if (exp >= 31 + bias) {
    /* number overflow */
    /* or f < 0 and (int)f == INT_MIN */
    num = 0x80000000;
  } else {
    E = exp - bias;
    M = frac | 0x800000;
    if (E > 23) {
      num = M << (E - 23);
    } else {
      /* whether sig is 1 or 0, round to zero */
      num = M >> (23 - E);
    }
  }

  return sig ? -num : num;
}
```

2.97

```
/*
 * float-i2f.c
 */
#include <stdio.h>
#include <assert.h>
#include <limits.h>
#include "float-i2f.h"

/*
 * Assume i > 0
 * calculate i's bit length
 *
 * e.g.
 * 0x3 => 2
 * 0xFF => 8
 * 0x80 => 8
 */
int bits_length(int i) {
  if ((i & INT_MIN) != 0) {
    return 32;
  }

  unsigned u = (unsigned)i;
  int length = 0;
  while (u >= (1<<length)) {
    length++;
  }
  return length;
}

/*
 * generate mask
 * 00000...(32-l) 11111....(l)
 *
 * e.g.
 * 3  => 0x00000007
 * 16 => 0x0000FFFF
 */
```

```
unsigned bits_mask(int l) {
  return (unsigned) -1 >> (32-l);
}

/*
 * Compute (float) i
 */
float_bits float_i2f(int i) {
  unsigned sig, exp, frac, rest, exp_sig /* except sig */, round
_part;
  unsigned bits, fbits;
  unsigned bias = 0x7F;

  if (i == 0) {
    sig = 0;
    exp = 0;
    frac = 0;
    return sig << 31 | exp << 23 | frac;
  }
  if (i == INT_MIN) {
    sig = 1;
    exp = bias + 31;
    frac = 0;
    return sig << 31 | exp << 23 | frac;
  }

  sig = 0;
  /* 2's complatation */
  if (i < 0) {
    sig = 1;
    i = -i;
  }

  bits = bits_length(i);
  fbits = bits - 1;
  exp = bias + fbits;

  rest = i & bits_mask(fbits);
  if (fbits <= 23) {
    frac = rest << (23 - fbits);
```

```
    exp_sig = exp << 23 | frac;
  } else {
    int offset = fbits - 23;
    int round_mid = 1 << (offset - 1);

    round_part = rest & bits_mask(offset);
    frac = rest >> offset;
    exp_sig = exp << 23 | frac;

    /* round to even */
    if (round_part < round_mid) {
      /* nothing */
    } else if (round_part > round_mid) {
      exp_sig += 1;
    } else {
      /* round_part == round_mid */
      if ((frac & 0x1) == 1) {
        /* round to even */
        exp_sig += 1;
      }
    }
  }

  return sig << 31 | exp_sig;
}
```

Machine-Level Representation of Programs

> To understand a program, you must become both the machine and the program.
>
> by Alan Perlis

3.1 - 3.57 visit book

3.58 - 3.75 visit here

test

code directory: `./code`

test way:

- assert means assert function from `<assert.h>`
- output means to watch code output to judge if it works right

solution	code file	test way
3.58	decode2/(decode.c, decode2.s. main.c)	assert
3.59	-----	------
3.60	loop/loop.s, loop2.c, main.c	assert
3.61	cread/(cread-alt.c, cread-alt.s, cread.s)	assert
3.62	-----	------
3.63	-----	------
3.64	store-ele/(store-ele.s, store-ele2.c, main.c)	assert
3.65	-----	------
3.66	sum-col/(sum-col.s, sum-col2.c, main.c)	assert
3.67	-----	------
3.68	-----	------
3.69	-----	------
3.70	-----	------
3.71	good-echo/good-echo.c	output
3.72	-----	------
3.73	3.73.c	assert
3.74	3.74.c	assert
3.75	-----	------

3.58

```c
/*
 * decode.c
 */
long decode(long x, long y, long z) {
  long tmp = y - z;
  return (tmp * x) ^ (tmp << 63 >> 63);
}
```

3.59

assume

$$ux = x + x_{63}2^{64}$$

$$uy = y + y_{63}2^{64}$$

multiple

$$ux \cdot uy = (x + x_{63}2^{64}) \cdot (y + y_{63}2^{64})$$

$$ux \cdot uy = x \cdot y + (x_{63}y + y_{63}x)2^{64} + x_{63}y_{63}2^{128}$$

2^128 overflows, don't care about it

$$x \cdot y = ux \cdot uy - (x_{63}y + y_{63}x)2^{64}$$

```
# void store_prod(int128_t* dest, int64_t x, int64_t y)
# dest in %rdi, x in %rsi, y in %rdx
store_prod:
  movq %rdx, %rax     # %rax = y
  cqto                # (int128_t)y, %rdx = (-1)y_63
  movq %rsi, %rcx     # %rcx = x

  # x >> 63, if x == 1, %rcx = -1; if x_63 == 0, %rcx = 0
  # %rcx = (-1)x_63
  sarq $63, %rcx

  # pay attention, imulq behaves differently according to param
number(1/2)
  imulq %rax, %rcx    # %rcx = y * -x_63
  imulq %rsi, %rdx    # %rdx = x * -y_63
  addq %rdx, %rcx     # %rcx = x * -y_63 + y * -x_63
  mulq %rsi           # %rdx:%rax <= ux * uy

  # lower 64 bits are same for x * y and ux * uy. ref (2.18) on
book
  # %rdx = ux * uy(high 64 bits) - (x_{63}y + y_{63}x)2^{64}
  addq %rcx, %rdx

  movq %rax, (%rdi)   # set lower 64bits
  movq %rdx, 8(%rdi)  # set higher 64bits
  ret
```

3.60

A.

val	reg
x	%rdi
n	%esi
result	%rax
mask	%rdx

B.

```
result = 0

mask = 1
```

C.

```
mask != 0
```

D.

```
mask = mask << n
```

E.

```c
/*
 * loop2.c
 */

long loop2(long x, int n) {
  long result = 0;
  long mask;
  for (mask = 1; mask != 0; mask <<= n) {
    result |= (x & mask);
  }
  return result;
}
```

3.61

function cread_alt should be

```
/*
 * cread-alt.c
 */
#include <stdio.h>
#include <assert.h>

long cread(long *xp) {
  return (xp ? *xp : 0);
}

long cread_alt(long *xp) {
  return (!xp ? 0 : *xp);
}

int main(int argc, char* argv[]) {
  long a = 0;
  assert(cread(&a) == cread_alt(&a));
  assert(cread(NULL) == cread_alt(NULL));
  return 0;
}
```

In book sample, compile function cread get

```
# long cread(long *xp)
# xp in %rdi
cread:
  movq (%rdi), %rax
  testq %rdi, %rdi
  movl $0, %edx
  cmove %rdx, %rax
  ret
```

compile cread_alt may get

```
# long cread_alt(long* xp)
# xp in %rdi
cread_alt:
  movl $0, %eax
  testq %rdi, %rdi
  cmovne (%rdi), %rax
```

3.62

```c
/*
 * 3.62.c
 */

/* Enumerated type creates set of contants numbered 0 and upward
 */
typedef enum { MODE_A, MODE_B, MODE_C, MODE_D, MODE_E } mode_t;

long switch3(long *p1, long *p2, mode_t action) {
  long result = 0;
  switch(action) {
    case MODE_A:
      *p2 = *p1;
      result = *p2;
      break;
    case MODE_B:
      *p1 = *p1 + *p2;
      result = *p1;
      break;
    case MODE_C:
      *p1 = 59;
      result = *p2;
      break;
    case MODE_D:
      *p1 = *p2;
      result = 27;
      break;
    case MODE_E:
      result = 27;
      break;
    default:
      result = 12;
      break;
  }
  return result;
}
```

3.63

```c
/*
 * 3.63.c
 */

long switch_prob(long x, long n) {
  long result = x;
  switch(n) {
    /* Fill in code here */
    case 60:
    case 62:
      result = x * 8;
      break;
    case 63:
      result = x >> 3;
      break;
    case 64:
      x = x << 4 - x;
    case 65:
      x = x * x;
    default:
      result = x + 0x4B;
  }
  return result;
}
```

3.64

A.

3.1 in book

```
T D[R][C];

&D[i][j] = Xd + L(C*i + j)

T means type, D means data, R means row, C means column

L means sizeof(T), Xd means address of D
```

similarly, in 3d array

```
TYPE D[R][S][T]

&D[i][j][k] = Xd + L(S*T*i + T*j + k)
```

B.

```
.section .data
.global A
A:
  .fill 3640/8, 8, 121 # fill data 121

.section .text
.global store_ele
# long store_ele(long i, long j, long k, long *dest)
# i in %rdi, j in %rsi, k in %rdx, dest in %rcx
store_ele:
  leaq (%rsi, %rsi, 2), %rax    # t1 = j*3
  leaq (%rsi, %rax, 4), %rax    # t1 = j*13
  movq %rdi, %rsi               # t2 = i
  salq $6, %rsi                 # t2 = i*64
  addq %rsi, %rdi               # t3 = i*65
  addq %rax, %rdi               # t3 = i*65 + j*13
  addq %rdi, %rdx               # t4 = i*65 + j*13 + k
  movq A(,%rdx,8), %rax         # t1 = *(A + 8*t4)
  movq %rax, (%rcx)             # *dest = t1
  movl $3640, %eax              # return 3640
  ret
```

base on comments,

```
S * T = 65
T = 13
8*R*S*T = 3640
```

so

```
R = 7
S = 5
T = 13
```

3.65

A.

```
&A[i][j] in %rdx
```

B.

```
&A[j][i] in %rax
```

C.

```
addq $8, %rdx # means A[i][j] -> A[i][j+1]

addq $120, %rax # means A[j][i] -> A[j+1][i], 120 == 8*M
```

M = 15

3.66

```
.section .text
.global sum_col
# long sum_col(long n, long A[NR(n)][NC(n)], long j)
# n in %rdi, A in %rsi, j in %rdx
sum_col:
  leaq 1(,%rdi,4), %r8         # t1 = n*4 + 1
  leaq (%rdi,%rdi,2), %rax     # t2 = n*3
  movq %rax, %rdi              # t3 = n*3
  testq %rax, %rax             # test n*3
  jle .L4                      # n*3 <= 0, jump .L4
  salq $3, %r8                 # t1 = t1*8 = 8*(n*4 + 1)
  leaq (%rsi,%rdx,8), %rcx     # t4 = j*8 + A
  movl $0, %eax                # t2 = 0
  movl $0, %edx                # t5 = 0
.L3:
  addq (%rcx), %rax            # t2 = *(t4) = *(A + j*8)
  addq $1, %rdx                # t5 = t5+1
  addq %r8, %rcx               # t4 = t1+t4 = A + j*8 + 8*(n*4 +
1)
  cmpq %rdi, %rdx              # cmp t5 & t3
  jne .L3                      # if t5 != n*3, loop
  rep
  ret
.L4:
  movl $0, %eax                # return 0
  ret
```

base on comments in asm code

```
cmpq %rdi, %rdx    # cmp t5 & t3
jne .L3            # if t5 != n*3, loop
```

t5 is var i, so `NR(n) == n*3`

```
leaq 1(,%rdi,4), %r8          # t1 = n*4 + 1
.....
salq $3, %r8                  # t1 = t1*8 = 8*(n*4 + 1)
.....
addq %r8, %rcx                # t4 = t1+t4 = A + j*8 + 8*(n*4 + 1)
```

in every loop, pointer move `8*(n*4 + 1)` bytes, so `NC(n) == n*4 + 1`

thanks gonglinyuan

3.67

```c
/*
 * 3.67.c
 */

typedef struct {
  long a[2];
  long *p;
} strA;

typedef struct {
  long u[2];
  long q;
} strB;

strB process(strA a) {
  strB r;
  r.u[0] = s.a[1];
  r.u[1] = s.a[0];
  r.q = *s.p;
  return r;
}

long eval(long x, long y, long z) {
  strA s;
  s.a[0] = x;
  s.a[1] = y;
  s.p = &z;
  strB r = process(s);
  return r.u[0] + r.u[1] + r.q;
}
```

```
# strB process(strA s)
# s in %rdi
process:
  movq %rdi, %rax
  movq 24(%rsp), %rdx
  movq (%rdx), %rdx
  movq 16(%rsp), %rcx
  movq %rcx, (%rdi)
  movq 8(%rsp), %rcx
  movq %rcx, 8(%rdi)
  movq %rdx, 16(%rdi)
  ret

# long eval(long x, long y, long z)
# x in %rdi, y in %rsi, z in %rdx
eval:
  subq $104, %rsp
  movq %rdx, 24(%rsp)
  leaq 24(%rsp), %rax
  movq %rdi, (%rsp)
  movq %rsi, 8(%rsp)
  movq %rax, 16(%rsp)
  leaq 64(%rsp), %rdi
  call process
  movq 72(%rsp), %rax
  addq 64(%rsp), %rax
  addq 80(%rsp), %rax
  addq $104, %rsp
  ret
```

A.

```
104  +-----------------+
     |                 |
     |                 |
     |                 |
     |                 |
     |                 |
     |                 |
     |                 |
     |                 |
 64  +-----------------+ <-- %rdi
     |                 |
     |                 |
     |                 |
     |                 |
     |                 |
     |                 |
 32  +-----------------+
     |        z        |
 24  +-----------------+
     |       &z        |
 16  +-----------------+
     |        y        |
  8  +-----------------+
     |        x        |
  0  +-----------------+ <-- %rsp
```

B.

eval pass a new address %rsp+64 to process

C.

process access s by %rsp+offset, not by %rdi

D.

eval pass address %rsp+64 to process, process store data from here as beginning, finially return this address

E.

```
104  +------------------+
     |                  |
     |                  |
     |                  |
     |                  |
     |                  |
     |                  |
 88  +------------------+
     |        z         |
 80  +------------------+
     |        x         |
 72  +------------------+
     |        y         |
 64  +------------------+ <-- %rdi(eval pass in)
     |                  |  \
     |                  |   -- %rax(process pass out)
     |                  |
     |                  |
     |                  |
     |                  |
 32  +------------------+
     |         z        |
 24  +------------------+
     |        &z        |
 16  +------------------+
     |         y        |
  8  +------------------+
     |         x        |
  0  +------------------+ <-- %rsp in eval
     |                  |
 -8  +------------------+ <-- %rsp in process
```

F.

caller find space and pass space address to callee, callee store data on this space area and return this address

3.68

```
# void setVal(str1* p, str2* q)
# p in %rdi, q in %rsi
setVal:
  # 8(%rsi) fetch q->t, int t is aligned by 4, so 4 < B <=8
  movslq 8(%rsi), %rax

  # 32(%rsi) fetch q->u, long u is aligned by 8
  # offset q->s is offset q->t + 4, so 24 < 12 + A*2 <= 32
  addq 32(%rsi), %rax

  # 184(%rdi) fetch p->v, long y is aligned by 8, so 176 < A*B*4
 <= 184
  movq %rax, 184(%rdi)
  ret
```

```
4 < B <= 8
5 < A <= 10
44 < A*B <= 46
```

only

A = 9

B = 5

3.69

```c
/*
 * 3.69.c
 */

typedef struct {
  int first;
  a_struct a[CNT];
  int last;
} b_struct;

void test(long i, b_struct *bp) {
  int n = bp->first + bp->last;
  a_struct *ap = &bp->a[i];
  ap->x[ap->idx] = n;
}
```

```
# void test(long i, b_struct *bp)
# i in %rdi, bp in %rsi
test:
  mov 0x120(%rsi), %ecx          # bp+0x120 fetch bp->last
  add (%rsi), %ecx               # bp->first + bp->last
  lea (%rdi,%rdi,4), %rax        # i*5
  lea (%rsi,%rax,8), %rax        # bp+i*40

  # ap = &bp->a[i] = bp+i*40+8, +8 means skip int first
  # so a_struct is aligned by 8, size is 40
  # check last instrction, %rdx here saves value ap->idx!!!
  # so in a_struct, idx is first element
  mov 0x8(%rax), %rdx

  movslq %ecx, %rcx              # n = bp->first + bp->last, conv
ert to long

  # save n to address 8*(ap->idx) + bp+i*40+0x8 + 0x8  (0x10)
  # bp+i*40+0x8 means ap
  # ap + 0x8 means &(ap->x)
  # ap + 0x8 + 8*(ap->idx) means &(ap-x[ap->idx])
  # second element of a_struct is x, x is an array of long
  # 8*(ap->idx) means idx is also long type
  mov %rcx, 0x10(%rax,%rdx,8)

  # size of a_struct is 40 and aligned by 8
  # so array x has 4 long elements
  # finally, a_struct is
  #   typedef struct {
  #     long idx,
  #     long x[4]
  #   } a_struct
  retq
```

A.

CNT = 4

B.

```
typedef struct {
  long idx,
  long x[4]
} a_struct
```

3.70

A.

val	offset
e1.p	0
e1.y	8
e2.x	0
e2.next	8

B.

16

C.

```
# void proc(union ele *up)
# up in %rdi
proc:
  # %rax = *(up+8), don't know it's next or y
  movq 8(%rdi), %rax

  # %rdx = *( *(up+8) ), %rax stands for a pointer
  # so *( *(up+8) ) means *(up->e2.next)
  movq (%rax), %rdx

  # %rdx = *( *(up->e2.next) )
  # %rdx is treated as a pointer
  # so %rdx stores *( *(up->e2.next).e1.p )
  movq (%rdx), %rdx

  # %rax stores *(up+8)
  # %rax is treated as a pointer
  # so %rax = *( up->e2.next ), stands for another union ele's a
ddress
  #
  # in subq, %rdx is a long number
  # *( *(up->e2.next)+8 ) must be a long number
  # so 8(%rax) means *(up->e2.next).e1.y
  subq 8(%rax), %rdx

  # %rdi never changes in previous instrctions
  # instrction below is the final assignment
  # so (%rdi) means up->e2.x
  movq %rdx, (%rdi)
  ret
```

base on comments

```c
/*
 * 3.70.c
 */

union ele {
  struct {
    long *p;
    long y;
  } e1;
  struct {
    long x;
    union ele *next;
  } e2;
};

void proc(union ele *up) {
  /* up->  = *(    ) -     ; */
  up->e2.x = *( *(up->e2.next).e1.p ) - *(up->e2.next).e1.y
}
```

3.71

```c
/*
 * good-echo.c
 */
#include <stdio.h>
#include <assert.h>
#define BUF_SIZE 12

void good_echo(void) {
  char buf[BUF_SIZE];
  while(1) {
    /* function fgets is interesting */
    char* p = fgets(buf, BUF_SIZE, stdin);
    if (p == NULL) {
      break;
    }
    printf("%s", p);
  }
  return;
}

int main(int argc, char* argv[]) {
  good_echo();
  return 0;
}
```

3.72

```c
/*
 * 3.72.c
 */

#include <alloca.h>

long aframe(long n, long idx, long *q) {
  long i;
  long **p = alloca(n * sizeof(long*));
  p[0] = &i;
  for (i = 1; i < n; i++) {
    p[i] = q;
  }
  return *p[idx];
}
```

```
# long aframe(long n, long idx, long *q)
# n in %rdi, idx in %rsi, q in %rdx
aframe:
  pushq %rbp
  movq %rsp, %rbp
  subq $16, %rsp
  leaq 30(,%rdi,8), %rax
  andq $-16, %rax
  subq %rax, %rsp
  leaq 15(%rsp), %r8
  andq $-16, %r8
```

A.

$$s_2 = s_1 - [(n * 8 + 30) \& 0XFFFFFFF0]$$

when n is odd

$$s_2 = s_1 - (n * 8 + 24)$$

when n is even

$$s_2 = s_1 - (n * 8 + 16)$$

B.

$$p = (s_2 + 15) \& 0XFFFFFFF0$$

the least multiple of 16 that greater than s2

C.

which	e1	n	s1
least	1	even	n%16==1
greatest	24	odd	n%16==0

least:

e1 can't be 0, if e1 == 0, p should equal to s2

when n is even, e1 + e2 == 16, if e2 is 15, e1 will be least that can reach, so s1 == n that n%16==1

greatest:

when n is odd, e1 + e2 == 24. if p == s2, e2 == 0, so e1 will be the greatest value 24 that can reach. s1 == n that n%16 == 0

D.

p is aligned by 16

s2 is the least multiple of 16 that preserve 8*n size space

3.73

```c
/*
 * 3.73.c
 */
#include <stdio.h>
#include <assert.h>

typedef enum {NEG, ZERO, POS, OTHER} range_t;

range_t find_range(float x) {
  __asm__(
      "vxorps %xmm1, %xmm1, %xmm1\n\t"
      "vucomiss %xmm1, %xmm0\n\t"
      "jp .P\n\t"
      "ja .A\n\t"
      "jb .B\n\t"
      "je .E\n\t"
      ".A:\n\t"
      "movl $2, %eax\n\t"
      "jmp .Done\n\t"
      ".B:\n\t"
      "movl $0, %eax\n\t"
      "jmp .Done\n\t"
      ".E:\n\t"
      "movl $1, %eax\n\t"
      "jmp .Done\n\t"
      ".P:\n\t"
      "movl $3, %eax\n\t"
      ".Done:\n\t"
      );
}

int main(int argc, char* argv[]) {
  range_t n = NEG, z = ZERO, p = POS, o = OTHER;
  assert(o == find_range(0.0/0.0));
  assert(n == find_range(-2.3));
  assert(z == find_range(0.0));
  assert(p == find_range(3.33));
  return 0;
}
```

3.74

```c
/*
 * 3.74.c
 */
#include <stdio.h>
#include <assert.h>

typedef enum {NEG, ZERO, POS, OTHER} range_t;

range_t find_range(float x) {
  __asm__(
      "vxorps %xmm1, %xmm1, %xmm1\n\t"
      "movq $1, %rax\n\t"
      "movq $2, %r8\n\t"
      "movq $0, %r9\n\t"
      "movq $3, %r10\n\t"
      "vucomiss %xmm1, %xmm0\n\t"
      "cmovaq %r8, %rax\n\t"
      "cmovbq %r9, %rax\n\t"
      "cmovpq %r10, %rax\n\t"
      );
}

int main(int argc, char* argv[]) {
  range_t n = NEG, z = ZERO, p = POS, o = OTHER;
  assert(o == find_range(0.0/0.0));
  assert(n == find_range(-2.3));
  assert(z == find_range(0.0));
  assert(p == find_range(3.33));
  return 0;
}
```

3.75

A.

param n	real	img
1	%xmm0	%xmm1
2	%xmm2	%xmm3
3	%xmm4	%xmm5
n	...	...

B.

return %xmm0 as real part, %xmm1 as img part

Processor Architecture

> The speed at which modern CPUs perform computations still blows my mind daily.
>
> by Markus Persson

4.1 - 4.44 visit book

4.45 - 4.59 visit here

yas simulation

This chapter focus on processor architecture and design a little simple processor and yas -- a assemble language -- designed running on it.

you can access the processor simulation code and simulation manual from csapp official site.

simulation code

simulator manual

I have saved the simulation code in directory chapter4/code/sim.

Highly recommend you read the manual and README in code to know how it works and how to test yas code.

test

code directory: `./code`

test way:

- assert means assert function from `<assert.h>`
- output means to watch code output to judge if it works right
- yas means using simulator test script to test simulator itself

solution	code file	test way
4.45	--	--
4.46	--	--
4.47	bubble-sort/bubble-sort-pointer.*	output, assert
4.48	bubble-sort/bubble-sort-pointer-3-cmove.*	output
4.49	bubble-sort/bubble-sort-pointer-1-cmove.*	output
4.50	switch/*	output
4.51	--	--
4.52	sim/seq/seq-full.hcl	yas
4.53	sim/pipe/pip-stall.hcl	yas
4.54	sim/pipe/pip-full.hcl	yas
4.55	sim/pipe/pip-nt.hcl	yas
4.56	sim/pipe/pip-btfnt.hcl	yas
4.57	sim/pipe/pip-lf.hcl	yas
4.58	sim/pipe/pip-1w.hcl	yas
4.59	--	--

4.45

```
subq $8, %rsp
movq REG, (%rsp)
```

A.

No

if REG is %rsp, we push %rsp-8 instead of %rsp into stack

B.

```
movq REG, -8(%rsp)
subq $8, %rsp
```

4.46

```
movq (%rsp), REG
addq $8, %rsp
```

A.

No

if REG is %rsp, `movq (%rsp), REG` pop the right value into %rsp, but `addq $8, %rsp` modify it

B.

```
addq $8, %rsp
movq 8(%rsp), REG
```

4.47

A.

Change into pointer version

```
/*
 * bubble-sort-pointer.c
 */

void bubble_p(long* data, long count) {
  long *i, *last;
  for (last = data+count-1; last > data; last--) {
    for (i = data; i < last; i++) {
      if (*(i+1) < *i) {
        /* swap adjacent elements */
        long t = *(i+1);
        *(i+1) = *i;
        *i = t;
      }
    }
  }
}
```

B.

Compile bubble-sort-pointer.c

```
gcc -S -Og bubble-sort-pointer.c
```

get gas file

```
.file    "bubble-sort-pointer.c"
    .text
    .globl    bubble_p
    .type    bubble_p, @function
bubble_p:
.LFB0:
    .cfi_startproc
    leaq    -8(%rdi,%rsi,8), %rsi
    jmp    .L2
.L4:
    movq    8(%rax), %rdx
    movq    (%rax), %rcx
    cmpq    %rcx, %rdx
    jge    .L3
    movq    %rcx, 8(%rax)
    movq    %rdx, (%rax)
.L3:
    addq    $8, %rax
    jmp    .L5
.L6:
    movq    %rdi, %rax
.L5:
    cmpq    %rsi, %rax
    jb    .L4
    subq    $8, %rsi
.L2:
    cmpq    %rdi, %rsi
    ja    .L6
    rep ret
    .cfi_endproc
.LFE0:
    .size    bubble_p, .-bubble_p
    .ident    "GCC: (Ubuntu 5.4.0-6ubuntu1~16.04.5) 5.4.0 201606
09"
    .section    .note.GNU-stack,"",@progbits
```

Change it into ys version

```
/* bubble-sort-pointer.ys */
```

```
.pos 0
    irmovq stack, %rsp
    call main
    halt

# Array of 4 elements
.align 8
data:
  .quad 0x0000000000000004
  .quad 0x0000000000000003
  .quad 0x0000000000000002
data_end:
  .quad 0x0000000000000001

main:
  irmovq data,%rdi
    irmovq data_end,%rsi
    call ysBubbleP
    ret

# long ys_bubble_p(long *data, long *end)
# data in %rdi, end in %rsi
ysBubbleP:
  jmp L2
L4:
  mrmovq 8(%rax), %r9
  mrmovq (%rax), %r10
  rrmovq %r9, %r8
  subq %r10, %r8
  jge L3
  rmmovq %r10, 8(%rax)
  rmmovq %r9, (%rax)
L3:
  irmovq $8, %r8
  addq %r8, %rax
  jmp L5
L6:
  rrmovq %rdi, %rax
L5:
  rrmovq %rsi, %r8
```

```
    subq %rax, %r8
    jg L4
    irmovq $8, %r8
    subq %r8, %rsi
  L2:
    rrmovq %rsi, %r8
    subq %rdi, %r8
    jg L6
      ret


  .pos 0x200
  stack:
```

main difference is second param of function is `long* end` instead of `long count` for convinient, `long* end` marks the position of last element.

using `yas` assemble it and `yis` run it, see output

```
../sim/misc/yas bubble-sort-pointer.ys
../sim/misc/yis bubble-sort-pointer.yo
Stopped in 117 steps at PC = 0x13.  Status 'HLT', CC Z=1 S=0 O=0
Changes to registers:
%rax:    0x0000000000000000    0x0000000000000020
%rsp:    0x0000000000000000    0x0000000000000200
%rsi:    0x0000000000000000    0x0000000000000018
%rdi:    0x0000000000000000    0x0000000000000018
%r9:    0x0000000000000000    0x0000000000000001
%r10:    0x0000000000000000    0x0000000000000002

Changes to memory:
0x0018:    0x0000000000000004    0x0000000000000001
0x0020:    0x0000000000000003    0x0000000000000002
0x0028:    0x0000000000000002    0x0000000000000003
0x0030:    0x0000000000000001    0x0000000000000004
0x01f0:    0x0000000000000000    0x0000000000000055
0x01f8:    0x0000000000000000    0x0000000000000013
```

initial array order is 4,3,2,1, now 1,2,3,4

4.48

a little easy changes.

```
/* bubble-sort-pointer.ys */
.pos 0
    irmovq stack, %rsp
    call main
    halt

# Array of 4 elements
.align 8
data:
  .quad 0x0000000000000004
  .quad 0x0000000000000003
  .quad 0x0000000000000002
data_end:
  .quad 0x0000000000000001

main:
  irmovq data,%rdi
    irmovq data_end,%rsi
    call ysBubbleP
    ret

# long ys_bubble_p(long *data, long *end)
# data in %rdi, end in %rsi
ysBubbleP:
  jmp L2
L4:
  mrmovq 8(%rax), %r9
  mrmovq (%rax), %r10
  rrmovq %r9, %r8
  subq %r10, %r8
##############################################################
# begin differences
##############################################################
  cmovl %r9, %r11
  cmovl %r10, %r9
  cmovl %r11, %r10
```

```
  rmmovq %r9, 8(%rax)
  rmmovq %r10, (%rax)
##############################################################
# end
##############################################################
  irmovq $8, %r8
  addq %r8, %rax
  jmp L5
L6:
  rrmovq %rdi, %rax
L5:
  rrmovq %rsi, %r8
  subq %rax, %r8
  jg L4
  irmovq $8, %r8
  subq %r8, %rsi
L2:
  rrmovq %rsi, %r8
  subq %rdi, %r8
  jg L6
    ret


.pos 0x200
stack:
```

run test, output

```
../sim/misc/yas bubble-sort-pointer-3-cmove.ys
../sim/misc/yis bubble-sort-pointer-3-cmove.yo
Stopped in 129 steps at PC = 0x13.  Status 'HLT', CC Z=1 S=0 O=0
Changes to registers:
%rax:	0x0000000000000000	0x0000000000000020
%rsp:	0x0000000000000000	0x0000000000000200
%rsi:	0x0000000000000000	0x0000000000000018
%rdi:	0x0000000000000000	0x0000000000000018
%r9:	0x0000000000000000	0x0000000000000002
%r10:	0x0000000000000000	0x0000000000000001
%r11:	0x0000000000000000	0x0000000000000001

Changes to memory:
0x0018:	0x0000000000000004	0x0000000000000001
0x0020:	0x0000000000000003	0x0000000000000002
0x0028:	0x0000000000000002	0x0000000000000003
0x0030:	0x0000000000000001	0x0000000000000004
```

4.49

it's a brilliant idea, 3 xor is a normal swap

if x = 9, y = 10

```
x = x ^ y  x:9^10; y:10
y = x ^ y  x:9^10; y:9
x = x ^ y  x:10; y:9
```

if change y after step 1

```
tmp = x    x:9; y:10; tmp:9  # store x
x = x ^ y  x:9^10; y:10
y = tmp    x:9^10; y:9       # set y to origin x
y = x ^ y  x:9^10; y:10
x = x ^ y  x:9; y:10
```

no swap happens. that's core in code below

```
/* bubble-sort-pointer.ys */
.pos 0
    irmovq stack, %rsp
    call main
    halt

# Array of 4 elements
.align 8
data:
  .quad 0x0000000000000004
  .quad 0x0000000000000003
  .quad 0x0000000000000002
data_end:
  .quad 0x0000000000000001

main:
  irmovq data,%rdi
    irmovq data_end,%rsi
    call ysBubbleP
```

```
    ret

# long ys_bubble_p(long *data, long *end)
# data in %rdi, end in %rsi
ysBubbleP:
  jmp L2
L4:
  mrmovq 8(%rax), %r9
  mrmovq (%rax), %r10
############################################
# begin differences
############################################
  rrmovq %r9, %r8
  rrmovq %r10, %r11
  xorq %r9, %r10
  subq %r11, %r8
  cmovge %r11, %r9
  xorq %r10, %r9
  xorq %r9, %r10
  rmmovq %r9, 8(%rax)
  rmmovq %r10, (%rax)
############################################
# end
############################################
  irmovq $8, %r8
  addq %r8, %rax
  jmp L5
L6:
  rrmovq %rdi, %rax
L5:
  rrmovq %rsi, %r8
  subq %rax, %r8
  jg L4
  irmovq $8, %r8
  subq %r8, %rsi
L2:
  rrmovq %rsi, %r8
  subq %rdi, %r8
  jg L6
    ret
```

```
.pos 0x200
stack:
```

test and output, watch memory changes:

```
../sim/misc/yas bubble-sort-pointer-1-cmove.ys
../sim/misc/yis bubble-sort-pointer-1-cmove.yo
Stopped in 141 steps at PC = 0x13.  Status 'HLT', CC Z=1 S=0 O=0
Changes to registers:
%rax:    0x0000000000000000    0x0000000000000020
%rsp:    0x0000000000000000    0x0000000000000200
%rsi:    0x0000000000000000    0x0000000000000018
%rdi:    0x0000000000000000    0x0000000000000018
%r9:    0x0000000000000000    0x0000000000000002
%r10:    0x0000000000000000    0x0000000000000001
%r11:    0x0000000000000000    0x0000000000000002

Changes to memory:
0x0018:    0x0000000000000004    0x0000000000000001
0x0020:    0x0000000000000003    0x0000000000000002
0x0028:    0x0000000000000002    0x0000000000000003
0x0030:    0x0000000000000001    0x0000000000000004
```

4.50

```
/* switch.ys */
.pos 0
    irmovq stack, %rsp
    call main
    halt

# Array of 4 elements
.align 8
array:
  .quad 0x0000000000000000
  .quad 0x0000000000000000
  .quad 0x0000000000000000
  .quad 0x0000000000000000

main:
  # test number 1, -1, 3, 5
  irmovq array, %r10

  irmovq $1,%rdi
    call switchv
  rmmovq %rax, (%r10)

  irmovq $-1,%rdi
    call switchv
  rmmovq %rax, 8(%r10)

  irmovq $3,%rdi
    call switchv
  rmmovq %rax, 16(%r10)

  irmovq $5,%rdi
    call switchv
  rmmovq %rax, 24(%r10)
    ret

table:
  .quad LD  # default branch
  .quad L0  # idx == 0
```

```
  .quad L1  # idx == 1
  .quad L2  # idx == 2
  .quad L3  # idx == 3
  .quad L4  # idx == 4
  .quad L5  # idx == 5

# long switchv(long idx)
# idx in %rdi
switchv:
  # contant number
  irmovq $8, %r8
  irmovq $0, %r10
  irmovq $1, %r11

  irmovq $0, %rax
  irmovq table, %rcx # table address
  rrmovq %rdi, %rdx
  subq %r8, %rdx
  jg def              # idx > 5
  subq %r10, %rdi
  jl def              # idx < 0
mul: # calculate 8 * %rdi
  subq %r10, %rdi
  je addr
  addq %r8, %rcx
  subq %r11, %rdi
  jmp mul
addr: # jump using table address
  addq %r8, %rcx
  mrmovq (%rcx), %rdi
  pushq %rdi
  ret
def: # default branch
  irmovq table, %rcx
  mrmovq (%rcx), %rdi
  pushq %rdi
  ret
L0:
  irmovq $0xaaa, %rax
  ret
```

```
L1:
  jmp LD
L2:
  jmp L5
L3:
  irmovq $0xccc, %rax
  ret
L4:
  jmp LD
L5:
  irmovq $0xbbb, %rax
  ret
LD:
  irmovq $0xddd, %rax
  ret

.pos 0x200
stack:
```

test function switchv in main, using idx 1,-1,3,5, store result into array

test and output, watch changes to memory:

```
../sim/misc/yas switch.ys
../sim/misc/yis switch.yo
Stopped in 133 steps at PC = 0x13.  Status 'HLT', CC Z=0 S=0 O=0
Changes to registers:
%rax:    0x0000000000000000    0x0000000000000bbb
%rcx:    0x0000000000000000    0x00000000000000e7
%rdx:    0x0000000000000000    0xfffffffffffffffd
%rsp:    0x0000000000000000    0x0000000000000200
%rdi:    0x0000000000000000    0x00000000000001a8
%r8:    0x0000000000000000    0x0000000000000008
%r11:    0x0000000000000000    0x0000000000000001

Changes to memory:
0x0000:    0x000000000200f430    0x0000000000000ddd
0x0008:    0x0000000038800000    0x0000000000000ddd
0x0010:    0x0000000000000000    0x0000000000000ccc
0x0018:    0x0000000000000000    0x0000000000000bbb
```

4.51

phase	iaddq V,rB
F	icode:ifun = M1[PC]; rA:rB = M1[PC+1]; valC = M8[PC+2]; valP = PC + 10;
D	valB = R[rB]
E	valE = valB + valC; set CC
M	
W	R[rB] = valE
PC	PC = valP

4.52

easy changes if you make 4.51 right.

check whole file `./chapter4/code/sim/seq/seq-full.hcl` with iaddq functionality

only watch changes with origin seq-full.hcl file

```
(cd ./chapter4/code/sim/seq; diff -u origin-seq-full.hcl seq-ful
l.hcl)
```

```
--- origin-seq-full.hcl    2017-11-09 02:57:43.671935966 +0000
+++ seq-full.hcl    2017-11-09 02:57:43.671935966 +0000
@@ -107,16 +107,16 @@

 bool instr_valid = icode in
     { INOP, IHALT, IRRMOVQ, IIRMOVQ, IRMMOVQ, IMRMOVQ,
-           IOPQ, IJXX, ICALL, IRET, IPUSHQ, IPOPQ };
+           IOPQ, IJXX, ICALL, IRET, IPUSHQ, IPOPQ, IIADDQ };

 # Does fetched instruction require a regid byte?
 bool need_regids =
     icode in { IRRMOVQ, IOPQ, IPUSHQ, IPOPQ,
-             IIRMOVQ, IRMMOVQ, IMRMOVQ };
+             IIRMOVQ, IRMMOVQ, IMRMOVQ, IIADDQ };

 # Does fetched instruction require a constant word?
 bool need_valC =
-    icode in { IIRMOVQ, IRMMOVQ, IMRMOVQ, IJXX, ICALL };
+    icode in { IIRMOVQ, IRMMOVQ, IMRMOVQ, IJXX, ICALL, IIADDQ };


 ################ Decode Stage    ##############################
#####

@@ -129,7 +129,7 @@

 ## What register should be used as the B source?
```

```
 word srcB = [
-    icode in { IOPQ, IRMMOVQ, IMRMOVQ } : rB;
+    icode in { IOPQ, IRMMOVQ, IMRMOVQ, IIADDQ } : rB;
     icode in { IPUSHQ, IPOPQ, ICALL, IRET } : RRSP;
     1 : RNONE;  # Don't need register
 ];
@@ -137,7 +137,7 @@
 ## What register should be used as the E destination?
 word dstE = [
     icode in { IRRMOVQ } && Cnd : rB;
-    icode in { IIRMOVQ, IOPQ} : rB;
+    icode in { IIRMOVQ, IOPQ, IIADDQ } : rB;
     icode in { IPUSHQ, IPOPQ, ICALL, IRET } : RRSP;
     1 : RNONE;  # Don't write any register
 ];
@@ -153,7 +153,7 @@
 ## Select input A to ALU
 word aluA = [
     icode in { IRRMOVQ, IOPQ } : valA;
-    icode in { IIRMOVQ, IRMMOVQ, IMRMOVQ } : valC;
+    icode in { IIRMOVQ, IRMMOVQ, IMRMOVQ, IIADDQ } : valC;
     icode in { ICALL, IPUSHQ } : -8;
     icode in { IRET, IPOPQ } : 8;
     # Other instructions don't need ALU
@@ -162,7 +162,7 @@
 ## Select input B to ALU
 word aluB = [
     icode in { IRMMOVQ, IMRMOVQ, IOPQ, ICALL,
-                 IPUSHQ, IRET, IPOPQ } : valB;
+                 IPUSHQ, IRET, IPOPQ, IIADDQ } : valB;
     icode in { IRRMOVQ, IIRMOVQ } : 0;
     # Other instructions don't need ALU
 ];
@@ -174,7 +174,7 @@
 ];

 ## Should the condition codes be updated?
-bool set_cc = icode in { IOPQ };
+bool set_cc = icode in { IOPQ, IIADDQ };
```

```
 ################# Memory Stage    ################################
#####
```

4.53

1. data hazard

something handled by data-forward must be handled by stall if no data-forward anymore. so when

```
d_srcA in { e_dstE, M_dstM, M_dstE, W_dstM, W_dstE } ||
d_srcB in { e_dstE, M_dstM, M_dstE, W_dstM, W_dstE }
```

data hazard happens, we have to insert bubble in phase E and stall phase F&D

load/use hazard

load/use hazard only happens when we use data-forward. no load/use hazard if no data-forward.

pay attention

- previous load/use toggle condition `d_srcA == E_dstM || d_srcB == E_dstM` now toggles data hazard.
- in the beginning, `d_srcA == RNONE == e_dstE == E_dstM == ...etc` , that's not data hazard

so we get:

```
situation: data_hazard
bool s_data_hazard =
  (
    (
      d_srcA != RNONE  &&
      d_srcA in { e_dstE, E_dstM, M_dstM, M_dstE, W_dstM, W_dstE
 }
    ) ||
    (
      d_srcB != RNONE  &&
      d_srcB in { e_dstE, E_dstM, M_dstM, M_dstE, W_dstM, W_dstE
 }
    )
  )
```

2. ret situation

keep same

```
situation: ret
bool s_ret = IRET in { D_icode, E_icode, M_icode };
```

3. jxx error

keep same

```
situation: jxx error
bool s_jxx_error = (E_icode == IJXX && !e_Cnd);
```

4. hazard composition

X means nothing to do; stall means stall; bubble means inserting bubble

num	data	ret	jxx	F	D	E	M	W
1	0	0	0	X	X	X	X	X
2	0	0	1	X	bubble	bubble	X	X
3	0	1	0	stall	bubble	X	X	X
4	1	0	0	stall	stall	bubble	X	X

situation 1: nothing happens, every thing is fine

situation 2: just jxx error, keep same with book

situation 3: just ret, keep same with book

situation 4: just data hazard, stall phase F and D, insert bubble in phase E, M and W keep same

what if two/three of them happen same time?

ret:

```
  +-----------+    +-----------+    +-----------+
M |           |  M |           |  M |    ret    |
  +-----------+    +-----------+    +-----------+
E |           |  E |    ret    |  E |    nop    |
  +-----------+    +-----------+    +-----------+
D |    ret    |  D |    nop    |  D |    nop    |
  +-----------+    +-----------+    +-----------+
        1                2                3
```

one of them

jxx error:

```
  +-----------+
M |           |
  +-----------+
E |    jxx    |
  +-----------+
D |           |
  +-----------+
```

when jxx error happens, E_icode must be jxx

data hazard:

```
              <--+
  +----------+   |
M |          |<--+
  +----------+   |
E |          |<--+
  +----------+   |
D |   xxx    +---+
  +----------+
```

when data hazard happens, D_icode is not sure, xxx means any instruction

composition 1: ret and jxx

keep same on book

composition 2: ret and data hazard

they two happens same time, so xxx must be ret

```
              <--+
  +----------+   |
M |          |<--+
  +----------+   |
E |          |<--+
  +----------+   |
D |   ret    +---+
  +----------+
```

when they happen same time, data hazard is prior to ret because if ret doesn't stall to avoid data hazard, we get wrong anwser with ISA

composition 3: jxx error and data hazard

```
               <--+
  +-----------+   |
M |           |<--+
  +-----------+   |
E |    jxx    |<--+
  +-----------+   |
D |    xxx    +---+
  +-----------+
```

when they two happens same time, jxx error is prior to data hazard because next 2 instructions is canceled when jxx error, xxx is canceled anymore.

composition 4: ret & jxx error & data hazard

```
               <--+
  +-----------+   |
M |           |<--+
  +-----------+   |
E |    jxx    |<--+
  +-----------+   |
D |    ret    +---+
  +-----------+
```

same like composition 3.

finally:

num	data	ret	jxx	F	D	E	M	W
1	0	0	0	X	X	X	X	X
2	0	0	1	X	bubble	bubble	X	X
3	0	1	0	stall	bubble	X	X	X
4	1	0	0	stall	stall	bubble	X	X
5	0	1	1	X	bubble	bubble	X	X
6	1	0	1	X	bubble	bubble	X	X
7	1	1	0	stall	stall	bubble	X	X
8	1	1	1	X	bubble	bubble	X	X

F:

- stall: (data || ret) && !jxx
- bubble: 0

D:

- stall: data && !jxx
- bubble: jxx || (!data && ret)

E:

- stall: 0
- bubble: jxx || data

M:

keep same

W:

keep same

finally:

check file `./chapter4/code/sim/pipe/pipe-stall.hcl`

watch changes with origin pipe-stall.hcl file

```
(cd ./chapter4/code/sim/pipe; diff -u origin-pipe-stall.hcl pipe
-stall.hcl)
```

```
--- origin-pipe-stall.hcl    2017-11-09 02:57:43.671935966 +0000
+++ pipe-stall.hcl    2017-11-09 02:57:43.671935966 +0000
@@ -303,40 +303,95 @@
 ];

 ################ Pipeline Register Control ####################
#####
+# situation: ret
+# bool s_ret = IRET in { D_icode, E_icode, M_icode };
+#
```

```
+# situation: jxx error
+# bool s_jxx_error = (E_icode == IJXX && !e_Cnd);
+#
+# situation: data_hazard
+# bool s_data_hazard =
+#   (
+#     (
+#       d_srcA != RNONE  &&
+#       d_srcA in { e_dstE, E_dstM, M_dstM, M_dstE, W_dstM, W_d
stE }
+#     ) ||
+#     (
+#       d_srcB != RNONE  &&
+#       d_srcB in { e_dstE, E_dstM, M_dstM, M_dstE, W_dstM, W_d
stE }
+#     )
+#   )

 # Should I stall or inject a bubble into Pipeline Register F?
 # At most one of these can be true.
-bool F_stall =
-    # Modify the following to stall the update of pipeline regi
ster F
-    0 ||
-    # Stalling at fetch while ret passes through pipeline
-    IRET in { D_icode, E_icode, M_icode };
+# bool F_stall = (s_ret || s_data_hazard) && !s_jxx_error;
+bool F_stall = (
+    (IRET in { D_icode, E_icode, M_icode }) ||
+    (
+      (
+        d_srcA != RNONE  &&
+        d_srcA in { e_dstE, E_dstM, M_dstM, M_dstE, W_dstM, W_d
stE }
+      ) ||
+      (
+        d_srcB != RNONE  &&
+        d_srcB in { e_dstE, E_dstM, M_dstM, M_dstE, W_dstM, W_d
stE }
+      )
```

```
+    )
+  ) &&
+  !(E_icode == IJXX && !e_Cnd);

 bool F_bubble = 0;

 # Should I stall or inject a bubble into Pipeline Register D?
 # At most one of these can be true.
-bool D_stall =
-    # Modify the following to stall the instruction in decode
-    0;
+# bool D_stall = s_data_hazard && !s_jxx_error;
+# bool D_bubble = s_jxx_error || (!s_data_hazard && s_ret)
+bool D_stall = (
+    (
+      d_srcA != RNONE  &&
+      d_srcA in { e_dstE, E_dstM, M_dstM, M_dstE, W_dstM, W_dst
E }
+    ) ||
+    (
+      d_srcB != RNONE  &&
+      d_srcB in { e_dstE, E_dstM, M_dstM, M_dstE, W_dstM, W_dst
E }
+    )
+  ) &&
+  !(E_icode == IJXX && !e_Cnd);

 bool D_bubble =
-    # Mispredicted branch
-    (E_icode == IJXX && !e_Cnd) ||
-    # Stalling at fetch while ret passes through pipeline
-    !(E_icode in { IMRMOVQ, IPOPQ } && E_dstM in { d_srcA, d_sr
cB }) &&
-    # but not condition for a generate/use hazard
-    !0 &&
-      IRET in { D_icode, E_icode, M_icode };
+  (E_icode == IJXX && !e_Cnd) ||
+  (
+    !(
+      (
```

```
+         d_srcA != RNONE  &&
+         d_srcA in { e_dstE, E_dstM, M_dstM, M_dstE, W_dstM, W_d
stE }
+       ) ||
+       (
+         d_srcB != RNONE  &&
+         d_srcB in { e_dstE, E_dstM, M_dstM, M_dstE, W_dstM, W_d
stE }
+       )
+     ) &&
+     (IRET in { D_icode, E_icode, M_icode })
+   );

 # Should I stall or inject a bubble into Pipeline Register E?
 # At most one of these can be true.
+# bool E_stall = 0;
+# bool E_bubble = s_jxx_error || s_data_hazard
 bool E_stall = 0;
 bool E_bubble =
-     # Mispredicted branch
-     (E_icode == IJXX && !e_Cnd) ||
-     # Modify the following to inject bubble into the execute st
age
-     0;
+   (E_icode == IJXX && !e_Cnd) ||
+   (
+     (
+       d_srcA != RNONE  &&
+       d_srcA in { e_dstE, E_dstM, M_dstM, M_dstE, W_dstM, W_dst
E }
+     ) ||
+     (
+       d_srcB != RNONE  &&
+       d_srcB in { e_dstE, E_dstM, M_dstM, M_dstE, W_dstM, W_dst
E }
+     )
+   );
+

 # Should I stall or inject a bubble into Pipeline Register M?
```

```
# At most one of these can be true.
```

4.54

almost same like 4.51 seq-full.hcl

check file `./chapter4/code/sim/pipe/pipe-full.hcl` with iaddq functionality

watch changes with origin pipe-full.hcl file

```
(cd ./chapter4/code/sim/pipe; diff -u origin-pipe-full.hcl pipe-
full.hcl)
```

```
--- origin-pipe-full.hcl    2017-11-09 02:57:43.671935966 +0000
+++ pipe-full.hcl    2017-11-09 02:57:43.671935966 +0000
@@ -158,7 +158,7 @@
 # Is instruction valid?
 bool instr_valid = f_icode in
     { INOP, IHALT, IRRMOVQ, IIRMOVQ, IRMMOVQ, IMRMOVQ,
-       IOPQ, IJXX, ICALL, IRET, IPUSHQ, IPOPQ };
+       IOPQ, IJXX, ICALL, IRET, IPUSHQ, IPOPQ, IIADDQ };

 # Determine status code for fetched instruction
 word f_stat = [
@@ -171,11 +171,11 @@
 # Does fetched instruction require a regid byte?
 bool need_regids =
     f_icode in { IRRMOVQ, IOPQ, IPUSHQ, IPOPQ,
-              IIRMOVQ, IRMMOVQ, IMRMOVQ };
+              IIRMOVQ, IRMMOVQ, IMRMOVQ, IIADDQ };

 # Does fetched instruction require a constant word?
 bool need_valC =
-    f_icode in { IIRMOVQ, IRMMOVQ, IMRMOVQ, IJXX, ICALL };
+    f_icode in { IIRMOVQ, IRMMOVQ, IMRMOVQ, IJXX, ICALL, IIADDQ
 };

 # Predict next value of PC
 word f_predPC = [
@@ -195,14 +195,14 @@

 ## What register should be used as the B source?
```

```
 word d_srcB = [
-    D_icode in { IOPQ, IRMMOVQ, IMRMOVQ  } : D_rB;
+    D_icode in { IOPQ, IRMMOVQ, IMRMOVQ, IIADDQ  } : D_rB;
     D_icode in { IPUSHQ, IPOPQ, ICALL, IRET } : RRSP;
     1 : RNONE;  # Don't need register
 ];

 ## What register should be used as the E destination?
 word d_dstE = [
-    D_icode in { IRRMOVQ, IIRMOVQ, IOPQ} : D_rB;
+    D_icode in { IRRMOVQ, IIRMOVQ, IOPQ, IIADDQ } : D_rB;
     D_icode in { IPUSHQ, IPOPQ, ICALL, IRET } : RRSP;
     1 : RNONE;  # Don't write any register
 ];
@@ -239,7 +239,7 @@
 ## Select input A to ALU
 word aluA = [
     E_icode in { IRRMOVQ, IOPQ } : E_valA;
-    E_icode in { IIRMOVQ, IRMMOVQ, IMRMOVQ } : E_valC;
+    E_icode in { IIRMOVQ, IRMMOVQ, IMRMOVQ, IIADDQ } : E_valC;
     E_icode in { ICALL, IPUSHQ } : -8;
     E_icode in { IRET, IPOPQ } : 8;
     # Other instructions don't need ALU
@@ -248,7 +248,7 @@
 ## Select input B to ALU
 word aluB = [
     E_icode in { IRMMOVQ, IMRMOVQ, IOPQ, ICALL,
-                 IPUSHQ, IRET, IPOPQ } : E_valB;
+                 IPUSHQ, IRET, IPOPQ, IIADDQ } : E_valB;
     E_icode in { IRRMOVQ, IIRMOVQ } : 0;
     # Other instructions don't need ALU
 ];
@@ -260,7 +260,7 @@
 ];

 ## Should the condition codes be updated?
-bool set_cc = E_icode == IOPQ &&
+bool set_cc = (E_icode == IOPQ || E_icode == IIADDQ) &&
     # State changes only during normal operation
     !m_stat in { SADR, SINS, SHLT } && !W_stat in { SADR, SINS,
```

```
  SHLT };
```

4.55

Problem mismatch with skeleton code, we follow the code here: **change J_YES to UNCOND**

one point is

```
M_icode == IJXX && M_ifun != UNCOND
```

means all jxx except jmp instruction

another important point is Mispredicted branch condition

origin

```
(E_icode == IJXX && !e_Cnd)
```

now

```
(E_icode == IJXX && E_ifun != UNCOND && e_Cnd)
```

if e_Cnd means misprediction

check file `./chapter4/code/sim/pipe/pipe-nt.hcl`

watch changes with origin pipe-nt.hcl

```
(cd ./chapter4/code/sim/pipe; diff -u origin-pipe-nt.hcl pipe-nt
.hcl)
```

```
--- origin-pipe-nt.hcl    2017-11-09 02:57:43.671935966 +0000
+++ pipe-nt.hcl    2017-11-09 02:57:43.671935966 +0000
@@ -139,7 +139,7 @@
 ## What address should instruction be fetched at
 word f_pc = [
     # Mispredicted branch.  Fetch at incremented PC
-    M_icode == IJXX && !M_Cnd : M_valA;
+    M_icode == IJXX && M_ifun != UNCOND && M_Cnd : M_valA;
```

```
     # Completion of RET instruction
     W_icode == IRET : W_valM;
     # Default: Use predicted value of PC
@@ -183,6 +183,7 @@
 # Predict next value of PC
 word f_predPC = [
     # BNT: This is where you'll change the branch prediction ru
le
+    f_icode == IJXX && f_ifun != UNCOND : f_valP;
     f_icode in { IJXX, ICALL } : f_valC;
     1 : f_valP;
 ];
@@ -273,7 +274,11 @@
     !m_stat in { SADR, SINS, SHLT } && !W_stat in { SADR, SINS,
 SHLT };

 ## Generate valA in execute stage
-word e_valA = E_valA;    # Pass valA through stage
+### pass branch address back by M_valA
+word e_valA = [
+    E_icode == IJXX && E_ifun != UNCOND : E_valC;
+    1 : E_valA;    # Pass valA through stage
+];

 ## Set dstE to RNONE in event of not-taken conditional move
 word e_dstE = [
@@ -343,7 +348,7 @@

 bool D_bubble =
     # Mispredicted branch
-    (E_icode == IJXX && !e_Cnd) ||
+    (E_icode == IJXX && E_ifun != UNCOND && e_Cnd) ||
     # Stalling at fetch while ret passes through pipeline
     # but not condition for a load/use hazard
     !(E_icode in { IMRMOVQ, IPOPQ } && E_dstM in { d_srcA, d_sr
cB }) &&
@@ -354,7 +359,7 @@
 bool E_stall = 0;
 bool E_bubble =
     # Mispredicted branch
```

```
-    (E_icode == IJXX && !e_Cnd) ||
+    (E_icode == IJXX && E_ifun != UNCOND && e_Cnd) ||
     # Conditions for a load/use hazard
     E_icode in { IMRMOVQ, IPOPQ } &&
      E_dstM in { d_srcA, d_srcB};
```

4.56

Problem mismatch with skeleton code, we follow the code here: **change J_YES to UNCOND**

almost same like 4.55. main differences are

- pay attention to whether valC is greater or lower than valP
- pass both valC and valP back by M registers because if misprediction happens we need judge jumping back to valC or valP

check file `./chapter4/code/sim/pipe/pipe-btfnt.hcl`

only watch changes with origin pipe-btfnt.hcl

```
(cd ./chapter4/code/sim/pipe; diff -u origin-pipe-btfnt.hcl pipe
-btfnt.hcl)
```

```
--- origin-pipe-btfnt.hcl    2017-11-09 02:57:43.667935995 +0000
+++ pipe-btfnt.hcl    2017-11-09 02:57:43.671935966 +0000
@@ -139,7 +139,10 @@
 ## What address should instruction be fetched at
 word f_pc = [
     # Mispredicted branch.  Fetch at incremented PC
-    M_icode == IJXX && !M_Cnd : M_valA;
+    # backward taken error
+    M_icode == IJXX && M_ifun != UNCOND && M_valE < M_valA && !
M_Cnd : M_valA;
+    # forward not-taken error
+    M_icode == IJXX && M_ifun != UNCOND && M_valE >= M_valA &&
M_Cnd : M_valE;
     # Completion of RET instruction
     W_icode == IRET : W_valM;
     # Default: Use predicted value of PC
@@ -183,6 +186,8 @@
 # Predict next value of PC
 word f_predPC = [
     # BBTFNT: This is where you'll change the branch prediction
 rule
+    f_icode == IJXX && f_ifun != UNCOND && f_valC < f_valP : f_
```

```
valC;
+    f_icode == IJXX && f_ifun != UNCOND && f_valC >= f_valP : f
_valP;
     f_icode in { IJXX, ICALL } : f_valC;
     1 : f_valP;
 ];
@@ -244,12 +249,15 @@
 # way to get valC into pipeline register M, so that
 # you can correct for a mispredicted branch.

+### pass valC by M_valE, pass valP by M_valA
+
 ## Select input A to ALU
 word aluA = [
     E_icode in { IRRMOVQ, IOPQ } : E_valA;
     E_icode in { IIRMOVQ, IRMMOVQ, IMRMOVQ } : E_valC;
     E_icode in { ICALL, IPUSHQ } : -8;
     E_icode in { IRET, IPOPQ } : 8;
+    E_icode in { IJXX } : E_valC;
     # Other instructions don't need ALU
 ];

@@ -258,6 +266,7 @@
     E_icode in { IRMMOVQ, IMRMOVQ, IOPQ, ICALL,
              IPUSHQ, IRET, IPOPQ } : E_valB;
     E_icode in { IRRMOVQ, IIRMOVQ } : 0;
+    E_icode in { IJXX } : 0;
     # Other instructions don't need ALU
 ];

@@ -343,7 +352,11 @@

 bool D_bubble =
     # Mispredicted branch
-    (E_icode == IJXX && !e_Cnd) ||
+    # backward taken error or forward not-taken error
+    (
+    (E_icode == IJXX && E_ifun != UNCOND && E_valC < E_valA &&
!e_Cnd) ||
+    (E_icode == IJXX && E_ifun != UNCOND && E_valC >= E_valA &&
```

```
 e_Cnd)
+    ) ||
     # BBTFNT: This condition will change
     # Stalling at fetch while ret passes through pipeline
     # but not condition for a load/use hazard
@@ -355,7 +368,11 @@
 bool E_stall = 0;
 bool E_bubble =
     # Mispredicted branch
-    (E_icode == IJXX && !e_Cnd) ||
+    # backward taken error or forward not-taken error
+    (
+    (E_icode == IJXX && E_ifun != UNCOND && E_valC < E_valA &&
!e_Cnd) ||
+    (E_icode == IJXX && E_ifun != UNCOND && E_valC >= E_valA &&
 e_Cnd)
+    ) ||
     # BBTFNT: This condition will change
     # Conditions for a load/use hazard
     E_icode in { IMRMOVQ, IPOPQ } &&
```

4.57

A.

consider load/use hazard

```
E_icode in { IMRMOVQ, IPOPQ } && E_dstM in { d_srcA, d_srcB };
```

situation	1	2	3	4
E_dstM == d_srcA	1	1	0	0
E_dstM == d_srcB	1	0	1	0

situation 4:

normal, no hazard happens

situation 1,2,3:

load/use hazard happens

if E_icode in { IMRMOVQ, IPOPQ }, then E_dstM must not be RNONE

consider situation 1 and 3, E_dstM == d_srcB, then d_srcB is not RNONE and must be used in phase E, so load-forward can't work. load-forward only work in situation 2!

consider all the instructions that d_srcA is not RNONE

instructions	d_srcA	valA used in phase E?	load-forward works?
rrmovq	rA	Y	N
rmmovq	rA	N	Y
opq	rA	Y	N
pushq	rA	N	Y
ret	rsp	N	N
popq	rsp	N	N

ret and popq can't work because d_srcA == d_srcB == %rsp, that's not in situation 2!!

finally, load/use only work in condition:

```
E_icode in { IMRMOVQ, IPOPQ } &&
(
  E_dstM == d_srcB ||
  (
    E_dstM == d_srcA && !(D_icode in { IRMMOVQ, IPUSHQ })
  )
);
```

B.

check file `./chapter4/code/sim/pipe/pipe-lf.hcl`

only watch changes with origin pipe-lf.hcl

```
(cd ./chapter4/code/sim/pipe; diff -u origin-pipe-lf.hcl pipe-lf
.hcl)
```

```
--- origin-pipe-lf.hcl    2017-11-09 02:57:43.671935966 +0000
+++ pipe-lf.hcl    2017-11-09 02:57:43.671935966 +0000
@@ -271,6 +271,7 @@
 ##    from memory stage when appropriate
 ## Here it is set to the default used in the normal pipeline
 word e_valA = [
+    E_icode in { IRMMOVQ, IPUSHQ } && E_srcA == M_dstM : m_valM;

     1 : E_valA;  # Use valA from stage pipe register
 ];

@@ -329,7 +330,13 @@
 bool F_stall =
     # Conditions for a load/use hazard
     ## Set this to the new load/use condition
-    0 ||
+    E_icode in { IMRMOVQ, IPOPQ } &&
+  (
+    E_dstM == d_srcB ||
+    (
```

```
+       E_dstM == d_srcA && !(D_icode in { IRMMOVQ, IPUSHQ })
+     )
+   ) ||
     # Stalling at fetch while ret passes through pipeline
     IRET in { D_icode, E_icode, M_icode };

@@ -338,15 +345,29 @@
 bool D_stall =
     # Conditions for a load/use hazard
     ## Set this to the new load/use condition
-    0;
+    E_icode in { IMRMOVQ, IPOPQ } &&
+   (
+    E_dstM == d_srcB ||
+    (
+      E_dstM == d_srcA && !(D_icode in { IRMMOVQ, IPUSHQ })
+    )
+   );

 bool D_bubble =
     # Mispredicted branch
     (E_icode == IJXX && !e_Cnd) ||
     # Stalling at fetch while ret passes through pipeline
     # but not condition for a load/use hazard
-    !(E_icode in { IMRMOVQ, IPOPQ } && E_dstM in { d_srcA, d_sr
cB }) &&
-      IRET in { D_icode, E_icode, M_icode };
+    !(
+        E_icode in { IMRMOVQ, IPOPQ } &&
+        (
+        E_dstM == d_srcB ||
+        (
+            E_dstM == d_srcA && !(D_icode in { IRMMOVQ, IPUSHQ
})
+            )
+        )
+    ) &&
+    IRET in { D_icode, E_icode, M_icode };

 # Should I stall or inject a bubble into Pipeline Register E?
```

```
 # At most one of these can be true.
@@ -356,7 +377,13 @@
     (E_icode == IJXX && !e_Cnd) ||
     # Conditions for a load/use hazard
     ## Set this to the new load/use condition
-    0;
+    E_icode in { IMRMOVQ, IPOPQ } &&
+    (
+        E_dstM == d_srcB ||
+        (
+            E_dstM == d_srcA && !(D_icode in { IRMMOVQ, IPUSHQ
})
+        )
+    );

 # Should I stall or inject a bubble into Pipeline Register M?
 # At most one of these can be true.
```

4.58

it's not a hard problem, focus on key points:

1. fetch popq instruction twice.
2. first time fetch popq, works like iaddq; second time fetch popq2, works like mrmovq
3. load/use condition should be popq2 not popq.

phase	popq rA	popq2 rA
works like	iadd $8, %rsp	mrmovq -8(%rsp), rA
F	valP = PC	valP = PC + 2
D	valB=R[rsp]	valB=R[rsp]
E	valE=valB+8	valE=valB-8
M		valM=M8[valE]
W	R[rsp]=valE	R[rA]=valM

check file `./chapter4/code/sim/pipe/pipe-1w.hcl`

only watch changes with origin pipe-1w.hcl

```
(cd ./chapter4/code/sim/pipe; diff -u origin-pipe-1w.hcl pipe-1w
.hcl)
```

```
--- origin-pipe-1w.hcl    2017-11-09 02:57:43.667935995 +0000
+++ pipe-1w.hcl    2017-11-09 02:57:43.671935966 +0000
@@ -157,6 +157,7 @@
 ## so that it will be IPOP2 when fetched for second time.
 word f_icode = [
     imem_error : INOP;
+    D_icode == IPOPQ : IPOP2;
     1: imem_icode;
 ];

@@ -169,7 +170,7 @@
 # Is instruction valid?
 bool instr_valid = f_icode in
```

```
     { INOP, IHALT, IRRMOVQ, IIRMOVQ, IRMMOVQ, IMRMOVQ,
-           IOPQ, IJXX, ICALL, IRET, IPUSHQ, IPOPQ };
+           IOPQ, IJXX, ICALL, IRET, IPUSHQ, IPOPQ, IPOP2 };

 # Determine status code for fetched instruction
 word f_stat = [
@@ -182,7 +183,7 @@
 # Does fetched instruction require a regid byte?
 bool need_regids =
     f_icode in { IRRMOVQ, IOPQ, IPUSHQ, IPOPQ,
-                 IIRMOVQ, IRMMOVQ, IMRMOVQ };
+                 IIRMOVQ, IRMMOVQ, IMRMOVQ, IPOP2 };

 # Does fetched instruction require a constant word?
 bool need_valC =
@@ -192,6 +193,7 @@
 word f_predPC = [
     f_icode in { IJXX, ICALL } : f_valC;
     ## 1W: Want to refetch popq one time
+    f_icode == IPOPQ : f_pc;
     1 : f_valP;
 ];

@@ -204,14 +206,14 @@
 ## What register should be used as the A source?
 word d_srcA = [
     D_icode in { IRRMOVQ, IRMMOVQ, IOPQ, IPUSHQ  } : D_rA;
-    D_icode in { IPOPQ, IRET } : RRSP;
+    D_icode in { IRET } : RRSP;
     1 : RNONE; # Don't need register
 ];

 ## What register should be used as the B source?
 word d_srcB = [
     D_icode in { IOPQ, IRMMOVQ, IMRMOVQ  } : D_rB;
-    D_icode in { IPUSHQ, IPOPQ, ICALL, IRET } : RRSP;
+    D_icode in { IPUSHQ, IPOPQ, IPOP2, ICALL, IRET } : RRSP;
     1 : RNONE;  # Don't need register
 ];
```

```
@@ -224,7 +226,7 @@

 ## What register should be used as the M destination?
 word d_dstM = [
-    D_icode in { IMRMOVQ, IPOPQ } : D_rA;
+    D_icode in { IMRMOVQ, IPOP2 } : D_rA;
     1 : RNONE;  # Don't write any register
 ];

@@ -255,7 +257,7 @@
 word aluA = [
     E_icode in { IRRMOVQ, IOPQ } : E_valA;
     E_icode in { IIRMOVQ, IRMMOVQ, IMRMOVQ } : E_valC;
-    E_icode in { ICALL, IPUSHQ } : -8;
+    E_icode in { ICALL, IPUSHQ, IPOP2 } : -8;
     E_icode in { IRET, IPOPQ } : 8;
     # Other instructions don't need ALU
 ];
@@ -263,7 +265,7 @@
 ## Select input B to ALU
 word aluB = [
     E_icode in { IRMMOVQ, IMRMOVQ, IOPQ, ICALL,
-             IPUSHQ, IRET, IPOPQ } : E_valB;
+             IPUSHQ, IRET, IPOPQ, IPOP2 } : E_valB;
     E_icode in { IRRMOVQ, IIRMOVQ } : 0;
     # Other instructions don't need ALU
 ];
@@ -292,13 +294,13 @@

 ## Select memory address
 word mem_addr = [
-    M_icode in { IRMMOVQ, IPUSHQ, ICALL, IMRMOVQ } : M_valE;
-    M_icode in { IPOPQ, IRET } : M_valA;
+    M_icode in { IRMMOVQ, IPUSHQ, ICALL, IMRMOVQ, IPOP2 } : M_v
alE;
+    M_icode in { IRET } : M_valA;
     # Other instructions don't need address
 ];

 ## Set read control signal
```

```
-bool mem_read = M_icode in { IMRMOVQ, IPOPQ, IRET };
+bool mem_read = M_icode in { IMRMOVQ, IRET, IPOP2 };
 
 ## Set write control signal
 bool mem_write = M_icode in { IRMMOVQ, IPUSHQ, ICALL };
@@ -350,7 +352,7 @@
 bool F_bubble = 0;
 bool F_stall =
     # Conditions for a load/use hazard
-    E_icode in { IMRMOVQ, IPOPQ } &&
+    E_icode in { IMRMOVQ, IPOP2 } &&
      E_dstM in { d_srcA, d_srcB } ||
     # Stalling at fetch while ret passes through pipeline
     IRET in { D_icode, E_icode, M_icode };
@@ -359,7 +361,7 @@
 # At most one of these can be true.
 bool D_stall =
     # Conditions for a load/use hazard
-    E_icode in { IMRMOVQ, IPOPQ } &&
+    E_icode in { IMRMOVQ, IPOP2 } &&
     E_dstM in { d_srcA, d_srcB };
 
 bool D_bubble =
@@ -367,7 +369,7 @@
     (E_icode == IJXX && !e_Cnd) ||
     # Stalling at fetch while ret passes through pipeline
     # but not condition for a load/use hazard
-    !(E_icode in { IMRMOVQ, IPOPQ } && E_dstM in { d_srcA, d_srcB }) &&
+    !(E_icode in { IMRMOVQ, IPOP2 } && E_dstM in { d_srcA, d_srcB }) &&
     # 1W: This condition will change
       IRET in { D_icode, E_icode, M_icode };
 
@@ -378,7 +380,7 @@
     # Mispredicted branch
     (E_icode == IJXX && !e_Cnd) ||
     # Conditions for a load/use hazard
-    E_icode in { IMRMOVQ, IPOPQ } &&
+    E_icode in { IMRMOVQ, IPOP2 } &&
```

```
      E_dstM in { d_srcA, d_srcB};

  # Should I stall or inject a bubble into Pipeline Register M?
```

4.59

4.47 is the better one

loop part in 4.47

```
L4:
  mrmovq 8(%rax), %r9
  mrmovq (%rax), %r10
  rrmovq %r9, %r8
  subq %r10, %r8
  jge L3
  rmmovq %r10, 8(%rax)
  rmmovq %r9, (%rax)
```

50% jge is right, run 5 instructions; 50% jge is wrong, run 7 instructions and 2 nop bubble. so Cycles Per Loop is 50% *5 + (7 + 2)* 50% = 7

loop part in 4.48

```
L4:
  mrmovq 8(%rax), %r9
  mrmovq (%rax), %r10
  rrmovq %r9, %r8
  subq %r10, %r8
  cmovl %r9, %r11
  cmovl %r10, %r9
  cmovl %r11, %r10
  rmmovq %r9, 8(%rax)
  rmmovq %r10, (%rax)
```

Cycles Per Loop is 9

loop part in 4.49

```
L4:
  mrmovq 8(%rax), %r9
  mrmovq (%rax), %r10
  rrmovq %r9, %r8
  rrmovq %r10, %r11
  xorq %r9, %r10
  subq %r11, %r8
  cmovge %r11, %r9
  xorq %r10, %r9
  xorq %r9, %r10
  rmmovq %r9, 8(%rax)
  rmmovq %r10, (%rax)
```

Cycles Per Loop is 11

Optimizing Program Performance

> We are all tasked to balance and optimize ourselves.
>
> by Mae Jemison

5.1 - 5.12 visit book

5.13 - 5.19 visit here

test

code directory: `./code`

test way:

- assert means assert function from `<assert.h>`
- output means to watch code output to judge if it works right

solution	code file	test way
5.13	5.13.c	assert
5.14	5.14.c	assert
5.15	5.15.c	assert
5.16	5.16.c	assert
5.17	5.17.c	assert
5.18	5.18.c	assert
5.19	5.19.c	assert

prof

prerequisite

- google gperftools

to 5.18 & 5.19, you can measure its performance.

```
(cd chapter5/code; make 5.18.prof)
(cd chapter5/code; make 5.18.prof)
```

5.13

A.

```
 +----+----+----+----+-----+-----+
 |%rbp|%rcx|%rax|%rbx|%xmm1|%xmm0|
 +----+----+----+----+-----+-----++----+
   +----|----|----|----------|--->|    |
   |    +----|----|----------|--->|load|      vmovad 0(%rbp,%rcx
,8),%xmm1
   |    |    |    |    +-----|----|    |
   |    |    |    |    |     |    +----+
   |    +----|----|----------|--->|    |
   |    |    |    |    |     |    |load|---+
   |    |    +----|----------|--->|    |   |
   |    |    |    |    |     |    +----+   |  vmulsd (%rax,%rcx,
8),%xmm1,%xmm0
   |    |    |    |    |     |    |    |<--+
   |    |    |    |    +-----|--->|mul |
   |    |    |    |    +-----|----|    |
   |    |    |    |    |     |    +----+
   |    |    |    |    +-----|--->|    |
   |    |    |    |    |     +--->|add |      vaddsd %xmm1,%xmm0
,%xmm0
   |    |    |    |    |     +----|    |
   |    |    |    |    |     |    +----+
   |    +----|----|----------|--->|    |
   |         |    |    |     |    |add |      addq $1, %rcx
   |    +----|----|----------|----|    |
   |    |    |    |    |     |    +----+
   |    +----|----|----------|--->|    |
   |    |    |    |    |     |    |cmp |---+  cmpq %rbx, %rcx
   |    |    |    +----|-----|--->|    |   |
   |    |    |    |    |     |    +----+   |
   |    |    |    |    |     |    |    |   |
   |    |    |    |    |     |    |jne |<--+  jne .L15
   |    |    |    |    |     |    |    |
   |    |    |    |    |     |    +----+
   v    v    v    v    v     v
 +----+----+----+----+-----+-----+
```

```
|%rbp|%rcx|%rax|%rbx|%xmm1|%xmm0|
+----+----+----+----+-----+-----+



     +----+                       +-----+
     |%rcx|                       |%xmm0|
     +----+                       +-----+
       |                            |
       |     +----+                 |  <--------- key path
       +---->|load|------+          |
       |     +----+      |          |
       |                 v          v
       |     +----+    +-+--+    +-+--+
       +---->|load|--->|mul |--->|add |
       |     +----+    +----+    +----+
       |                            |
       |                            |
       v                            |
     +----+                         |
     |add |                         |
     +----+                         |
       |                            |
       v                            v
     +-+--+                       +-----+
     |%rcx|                       |%xmm0|
     +----+                       +-----+
```

B.

5-12

float add cell, CPE is 3.0

C.

5-12

long add cell, 1.0

D.

only float add on key path

```
/*
 * 5.13.c
 */
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include "./lib/vec.h"

/* inner product. accumulate in temporary */
void inner4(vec_ptr u, vec_ptr v, data_t *dest) {
  long i;
  long length = vec_length(u);
  data_t *udata = get_vec_start(u);
  data_t *vdata = get_vec_start(v);
  data_t sum = (data_t) 0;

  for (i = 0; i < length; i++) {
    sum = sum + udata[i] * vdata[i];
  }
  *dest = sum;
}

int main(int argc, char* argv[]) {
  vec_ptr u = new_vec(4);
  vec_ptr v = new_vec(4);

  data_t *arr = (data_t*) malloc(sizeof(data_t) * 4);
  arr[0] = 0;
  arr[1] = 1;
  arr[2] = 2;
  arr[3] = 3;

  set_vec_start(u, arr);
  set_vec_start(v, arr);

  data_t res;
  inner4(u, v, &res);
```

```
  assert(res == 1+4+9);
  return 0;
}
```

5.14

```
+----+                          +----+
|sum |                          | i  |
+----+                          +----+
  |                               |
  v                               |
+----+       +----+  +----+       |
|add |<------|mul |<-|load|<------+
+----+       +----+  +----+       |
  |                               |
  v                               |
+----+       +----+  +----+       |
|add |<------|mul |<-|load|<------+
+----+       +----+  +----+       |
  |                               |
  v                               |
+----+       +----+  +----+       |
|add |<------|mul |<-|load|<------+
+----+       +----+  +----+       |
  |                               |
  v                               |
+----+       +----+  +----+       |
|add |<------|mul |<-|load|<------+
+----+       +----+  +----+       |
  |                               |
  v                               |
+----+       +----+  +----+       |
|add |<------|mul |<-|load|<------+
+----+       +----+  +----+       |
  |                               |
  v                               |
+----+       +----+  +----+       |
|add |<------|mul |<-|load|<------+
+----+       +----+  +----+       |
  |                               |
  |                               v
  |                             +----+
  |                             |add |
  |  <------- key path          +----+
```

```
    |                               |
    |                               |
    v                               v
 +----+                          +-+--+
 |sum |                          | i  |
 +----+                          +----+
```

A.

every element has 6 long/float add

element count is n/6

so n/6 * 6 = n

CPE bound == 1.0

B.

same like A

```
/*
 * 5.14.c
 */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#include "./lib/vec.h"

#define LEN 24

/* inner product. accumulate in temporary */
void inner4(vec_ptr u, vec_ptr v, data_t *dest) {
  long i;
  long length = vec_length(u);
  data_t *udata = get_vec_start(u);
  data_t *vdata = get_vec_start(v);
  data_t sum = (data_t) 0;

  for (i = 0; i < length-6; i+=6) {
```

```
    sum = sum + udata[i] * vdata[i] +
      udata[i+1] * vdata[i+1] +
      udata[i+2] * vdata[i+2] +
      udata[i+3] * vdata[i+3] +
      udata[i+4] * vdata[i+4] +
      udata[i+5] * vdata[i+5];
  }
  for(; i < length; i++) {
    sum = sum + udata[i] * vdata[i];
  }
  *dest = sum;
}

int main(int argc, char* argv[]) {
  vec_ptr u = new_vec(LEN);
  vec_ptr v = new_vec(LEN);

  data_t *arr = (data_t*) malloc(sizeof(data_t) * LEN);
  memset(arr, 0, sizeof(data_t) * LEN);
  arr[0] = 0;
  arr[1] = 1;
  arr[2] = 2;
  arr[3] = 3;

  set_vec_start(u, arr);
  set_vec_start(v, arr);

  data_t res;
  inner4(u, v, &res);

  assert(res == 1+4+9);
  return 0;
}
```

5.15

maybe

1. float mul capatity limit(less than 6)
2. register renaming limit

```
/*
 * 5.15.c
 */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#include "./lib/vec.h"

#define LEN 24

/* inner product. accumulate in temporary */
void inner4(vec_ptr u, vec_ptr v, data_t *dest) {
  long i;
  long length = vec_length(u);
  data_t *udata = get_vec_start(u);
  data_t *vdata = get_vec_start(v);
  data_t sum = (data_t) 0;
  data_t sum1 = (data_t) 0;
  data_t sum2 = (data_t) 0;
  data_t sum3 = (data_t) 0;
  data_t sum4 = (data_t) 0;
  data_t sum5 = (data_t) 0;

  for (i = 0; i < length-6; i+=6) {
    sum = sum + udata[i] * vdata[i];
    sum1 = sum1 + udata[i+1] * vdata[i+1];
    sum2 = sum2 + udata[i+2] * vdata[i+2];
    sum3 = sum3 + udata[i+3] * vdata[i+3];
    sum4 = sum4 + udata[i+4] * vdata[i+4];
    sum5 = sum5 + udata[i+5] * vdata[i+5];
  }
  for(; i < length; i++) {
```

```
    sum = sum + udata[i] * vdata[i];
  }
  *dest = sum + sum1 + sum2 + sum3 + sum4 + sum5;
}

int main(int argc, char* argv[]) {
  vec_ptr u = new_vec(LEN);
  vec_ptr v = new_vec(LEN);

  data_t *arr = (data_t*) malloc(sizeof(data_t) * LEN);
  memset(arr, 0, sizeof(data_t) * LEN);
  arr[0] = 0;
  arr[11] = 1;
  arr[2] = 2;
  arr[23] = 3;

  set_vec_start(u, arr);
  set_vec_start(v, arr);

  data_t res;
  inner4(u, v, &res);

  assert(res == 1+4+9);
  return 0;
}
```

5.16

```c
/*
 * 5.16.c
 */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#include "./lib/vec.h"

#define LEN 24

/* inner product. accumulate in temporary */
void inner4(vec_ptr u, vec_ptr v, data_t *dest) {
  long i;
  long length = vec_length(u);
  data_t *udata = get_vec_start(u);
  data_t *vdata = get_vec_start(v);
  data_t sum = (data_t) 0;

  for (i = 0; i < length-6; i+=6) {
    sum = sum +
      (
        udata[i] * vdata[i] +
        udata[i+1] * vdata[i+1] +
        udata[i+2] * vdata[i+2] +
        udata[i+3] * vdata[i+3] +
        udata[i+4] * vdata[i+4] +
        udata[i+5] * vdata[i+5]
      );
  }
  for(; i < length; i++) {
    sum = sum + udata[i] * vdata[i];
  }
  *dest = sum;
}

int main(int argc, char* argv[]) {
  vec_ptr u = new_vec(LEN);
```

```
  vec_ptr v = new_vec(LEN);

  data_t *arr = (data_t*) malloc(sizeof(data_t) * LEN);
  memset(arr, 0, sizeof(data_t) * LEN);
  arr[0] = 0;
  arr[11] = 1;
  arr[2] = 2;
  arr[23] = 3;

  set_vec_start(u, arr);
  set_vec_start(v, arr);

  data_t res;
  inner4(u, v, &res);

  assert(res == 1+4+9);
  return 0;
}
```

5.17

```c
/*
 * 5.17.c
 */
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>

void* basic_memset(void *s, int c, size_t n) {
  size_t cnt = 0;
  unsigned char *schar = s;
  while (cnt < n) {
    *schar++ = (unsigned char) c;
    cnt++;
  }
  return s;
}

/*
 * K = sizeof(unsigned long)
 * cs store K chars for memset
 */
void* effective_memset(void *s, unsigned long cs, size_t n) {
  /* align to K */
  size_t K = sizeof(unsigned long);
  size_t cnt = 0;
  unsigned char *schar = s;
  while (cnt < n) {
    if ((size_t)schar % K == 0) {
      break;
    }
    *schar++ = (unsigned char)cs;
    cnt++;
  }

  /* set K chars one time */
  unsigned long *slong = (unsigned long *)schar;
```

```
  size_t rest = n - cnt;
  size_t loop = rest / K;
  size_t tail = rest % K;

  for (size_t i = 0; i < loop; i++) {
    *slong++ = cs;
  }

  /* pad the tail part */
  schar = (unsigned char *)slong;
  for (size_t i = 0; i < tail; i++) {
    *schar++ = (unsigned char)cs;
  }
  return s;
}


int main(int argc, char* argv[]) {
  size_t space = sizeof(char) * 65537;
  // careful! malloc SIZE_MAX size memory will make sys slow
  // or crash down
  // size_t space = SIZE_MAX;

  void* basic_space = malloc(space);
  void* effective_space = malloc(space);

  int basic_fill = 0xFF;
  unsigned long effective_fill = ~0;

  basic_memset(basic_space, basic_fill, space);
  effective_memset(effective_space, effective_fill, space);

  assert(memcmp(basic_space, effective_space, space) == 0);

  free(basic_space);
  free(effective_space);
  return 0;
}
```

5.18

try to write 6*3a version for function holy.

this solution is not fully tested on my machine, sorry for that :(

```
uname -p
```

> Intel(R) Core(TM) i5-3210M CPU @ 2.50GHz

if your machine core is Intel Core i7 Haswell like book, you can try install google gperftools for profile

```
(cd chapter5/code; make 5.18.prof)
```

loop more times is a good idea :)

```
/*
 * 5.18.c
 */
#include <stdio.h>
#include <assert.h>

/* calculate a0 + a1*x + a2*x^2 + ... + an*x^n */
double poly(double a[], double x, long degree) {
  long i;
  double result = a[0];
  double xpwr = x;
  for (i = 1; i <= degree; i++) {
    result += a[i] * xpwr;
    xpwr = x * xpwr;
  }
  return result;
}

/* version 6*3a */
double poly_6_3a(double a[], double x, long degree) {
  long i = 1;
  double result = a[0];
```

```
  double result1 = 0;
  double result2 = 0;

  double xpwr = x;
  double xpwr1 = x * x * x;
  double xpwr2 = x * x * x * x * x;

  double xpwr_step = x * x * x * x * x * x;
  for (; i <= degree - 6; i+=6) {
    result = result + (a[i]*xpwr + a[i+1]*xpwr*x);
    result1 = result1 + (a[i+2]*xpwr1 + a[i+3]*xpwr1*x);
    result2 = result2 + (a[i+4]*xpwr2 + a[i+5]*xpwr2*x);

    xpwr *= xpwr_step;
    xpwr1 *= xpwr_step;
    xpwr2 *= xpwr_step;
  }

  for (; i <= degree; i++) {
    result = result + a[i]*xpwr;
    xpwr *= x;
  }

  return result + result1 + result2;
}

/* apply horner's method */
double polyh(double a[], double x, long degree) {
  long i;
  double result = a[degree];
  for (i = degree-1; i >= 0; i--) {
    result = a[i] + x*result;
  }
  return result;
}

#define LOOP 1000
#define LEN 1000

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

```
  double a[10 + 1] = { 0, 1, 1, 1, 1, 1,
                            1, 1, 1, 1, 1};
  double x = 2;
  long degree = 10;

  assert(poly(a, x, degree) == 2+4+8+16+32+(2+4+8+16+32)*32);
  assert(poly_6_3a(a, x, degree) == 2+4+8+16+32+(2+4+8+16+32)*32
);
  assert(poly(a, x, degree) == polyh(a, x, degree));

  x = 1;
  degree = LEN;
  double b[LEN + 1];

  for (int c = 0; c < LOOP; c++) {
    poly(b, x, degree);
  }
  for (int c = 0; c < LOOP; c++) {
    poly_6_3a(b, x, degree);
  }
  for (int c = 0; c < LOOP; c++) {
    polyh(b, x, degree);
  }
  return 0;
}
```

5.19

this solution's performance is not fully meassured.

if you're interested, you can try install google's gperftools for profile

```
(cd chapter5/code; make 5.19.prof)
```

loop more times is a good idea :)

```
/*
 * 5.19.c
 */
#include <stdio.h>
#include <assert.h>

void psum1a(float a[], float p[], long n) {
  long i;
  float last_val, val;
  last_val = p[0] = a[0];
  for (i = 1; i < n; i++) {
    val = last_val + a[i];
    p[i] = val;
    last_val = val;
  }
}

/* version 4*1a */
void psum_4_1a(float a[], float p[], long n) {
  long i;
  float val, last_val;
  float tmp, tmp1, tmp2, tmp3;
  last_val = p[0] = a[0];

  for (i = 1; i < n - 4; i++) {
    tmp = last_val + a[i];
    tmp1 = tmp + a[i+1];
    tmp2 = tmp1 + a[i+2];
    tmp3 = tmp2 + a[i+3];
```

```
    p[i] = tmp;
    p[i+1] = tmp1;
    p[i+2] = tmp2;
    p[i+3] = tmp3;

    /* key point */
    last_val = last_val + (a[i] + a[i+1] + a[i+2] + a[i+3]);
  }

  for (; i < n; i++) {
    last_val += a[i];
    p[i] = last_val;
  }
}

#define LOOP 1000
#define LEN  1000

int main(int argc, char* argv[]) {
  float a[5] = { 1, 2, 3, 4, 5 };
  float p[5];
  psum1a(a, p, 5);
  assert(p[4] == 15);

  float q[5];
  psum_4_1a(a, q, 5);
  assert(q[4] == 15);

  /* for prof */
  for (int i = 0; i < LOOP; i++) {
    float s[LEN];
    float d[LEN];
    psum1a(s, d, LEN);
    psum_4_1a(s, d, LEN);
  }
  return 0;
}
```

The Memory Hierarchy

> The more storage you have, the more stuff you accumulate.
>
> by Alexis Stewart

6.1 - 6.21 visit book

6.22 - 6.46 visit here

test

code directory: `./code`

test way:

- assert means assert function from `<assert.h>`
- output means to watch code output to judge if it works right

solution	code file	test way
6.22	----	----
6.23	----	----
6.24	----	----
6.25	----	----
6.26	----	----
6.27	----	----
6.28	----	----
6.29	----	----
6.30	----	----
6.31	----	----
6.32	----	----
6.33	----	----
6.34	----	----
6.35	----	----
6.36	----	----
6.37	----	----
6.38	----	----
6.39	----	----
6.40	----	----
6.41	----	----
6.42	----	----
6.43	----	----
6.44	----	----
6.45	transpose.c	no test, mainly measure performance
6.46	convert.c	no test, mainly measure performance

6.22

assume

Bits Per Track

```
bpt = x * r * K
```

Track Count

```
tc = (1-x) * r * M
```

M, K are constant

so

Bit Count

```
bc = K * M * r^2 * (1-x) * x
```

when x == 1/2, bc is maximum

6.23

T_avg_seek = 4ms

T_avg_rotation = 1/2 *1/15000* 60s/min * 1000ms/s = 2ms

T_avg_transfer = 1/15000 *1/800* 60s/min * 1000ms/s = 0.005ms

so

T_access = 6.005ms

6.24

almost same like problem 6.4 on book

A.

best case: blocks are mapped sequential and on same cylinder. just seek data once.

```
T_avg_seek = 4ms
T_avg_rotation = 2ms
```

file size 2MB, block size 512B, block count 2MB/512B = 4000

Block Per Track = 1000, so we need rotate 4 loop to read all data

```
T_transfer = T_rotation = T_max_rotation * 4 = 16ms
```

so

```
T_access = 22ms
```

B.

worse case: blocks are random.

```
T_access = 4000 * (T_avg_seek + T_avg_rotation) = 24s
```

6.25

m	c	B	E	S	t	s	b
32	1024	4	4	64	24	6	2
32	1024	4	256	1	30	0	2
32	1024	8	1	128	22	7	3
32	1024	8	128	1	29	0	3
32	1024	32	1	32	22	5	5
32	1024	32	4	8	24	3	5

6.26

m	c	B	E	S	t	s	b
32	2048	8	1	256	21	8	3
32	2048	4	4	128	23	7	2
32	1024	2	8	64	25	6	1
32	1024	32	2	16	23	4	5

6.27

A.

t = 0x45 = 0b01000101, s = 0b001, b = xx(xx is 00/01/10/11)

address may be

```
01000101 001 xx

format
0 1000 1010 01xx
```

address range: 0x08A4 - 0x08A7

t = 0x38

address range: 0x0704 - 0x0707

B.

0x1238 - 0x123B

6.28

same like 6.27

A.

None

B.

0x18F0 - 0x18F3

0x00B0 - 0x00B3

C.

0x0E34 - 0x0E37

D.

0x1BDC - 0x1BDF

6.29

A.

```
  11   10    9    8    7    6    5    4    3    2    1   0
+----+----+----+----+----+----+----+----+----+----+----+----+
| CT | CT | CT | CT | CT | CT | CT | CT | CI | CI | CO | CO |
+----+----+----+----+----+----+----+----+----+----+----+----+
```

B.

read/write	addr	hit?	value(or unknown)
read	0x834	No	-
write	0x836	Yes	unknown
read	0xFFD	Yes	0xC0

6.30

A.

C = E *B* S = 128B

B.

```
  12   11   10    9    8    7    6    5    4    3    2    1   0
+----+----+----+----+----+----+----+----+----+----+----+----+---
-+
| CT | CT | CT | CT | CT | CT | CT | CT | CI | CI | CI | CO | CO
 |
+----+----+----+----+----+----+----+----+----+----+----+----+---
-+
```

6.31

A.

```
  12   11   10    9    8    7    6    5    4    3    2    1   0
+----+----+----+----+----+----+----+----+----+----+----+----+----+
|  0 |  0 |  1 |  1 |  1 |  0 |  0 |  0 |  1 |  1 |  0 |  1 |  0 |
+----+----+----+----+----+----+----+----+----+----+----+----+----+
```

B.

param	value
CO	0x02
CI	0x06
CT	0x38
hit?	Yes
return	0xEB

6.32

A.

```
  12   11   10    9    8    7    6    5    4    3    2    1   0
+----+----+----+----+----+----+----+----+----+----+----+----+---
-+
|  1 |  0 |  1 |  1 |  0 |  1 |  1 |  1 |  0 |  1 |  0 |  0 |  0
 |
+----+----+----+----+----+----+----+----+----+----+----+----+---
-+
```

B.

param	value
CO	0x00
CI	0x02
CT	0xB7
hit?	No
return	--

6.33

same like 6.27

0x1788 - 0x178B

0x16C8 - 0x16CB

6.34

src:

	c0	c1	c2	c3
r0	m	m	h	m
r1	m	h	m	h
r2	m	m	h	m
r3	m	h	m	h

dst:

	c0	c1	c2	c3
r0	m	m	m	m
r1	m	m	m	m
r2	m	m	m	m
r3	m	m	m	m

6.35

src:

	c0	c1	c2	c3
r0	m	h	h	h
r1	m	h	h	h
r2	m	h	h	h
r3	m	h	h	h

dst:

	c0	c1	c2	c3
r0	m	h	h	h
r1	m	h	h	h
r2	m	h	h	h
r3	m	h	h	h

6.36

```c
int x[2][128];
int i;
int sum = 0;

for (i = 0; i < 128; i++) {
  sum += x[0][i] * x[1][i];
}
```

A.

C = 512, E = 1, B = 16, S = 32

total read count: 2 * 128

x[0][i] address: i*4

x[1][i] address: (128+i)*4 = 512 + i*4

so x[0][i] and x[1][i] are cached into same block.

miss rate 100%

B.

C = 1024, E = 1, B = 16, S = 64

sizeof(x) == 2 *128* 4 == C

whole array can be cached.

miss rate is depended on block size B.

B = 16, sizeof(int) = 4, so

miss rate is 25%

C.

C = 512, E = 2, B = 16, S = 16

total read count: 2 * 128

x[0][i] address: i*4

x[1][i] address: (128+i)*4 = 512 + i*4

so x[0][i] and x[1][i] are cached into different block in same set.

in first half, all elements can be cached. miss rate is 25%.

```
for (i = 0; i < 64; i++)
  sum += x[0][i] * x[1][i];
```

in second half

```
for (i = 64; i < 128; i++)
  sum += x[0][i] * x[1][i];
```

x[0][i] is not in cache. according to LRU strategy, cache x[0][i] into the same block with x[0][i-64], cache x[1][i] into the same block with x[1][i-64]. miss rate is 25%.

finally, miss rate is still 25%.

D.

No

if B is still 16, sizeof(int) = 4, block can only cache 4 int one time.

read int first time toggle memory cache, miss; next 3 time read hit.

25% miss rate is lowest.

E.

Yes

assume B = 32, block cache 8 int at one time.

only 1 miss in 8 time int read.

miss rate can be 12.5%.

6.37

```c
/*
 * 6.37.c
 */

typedef int array_t[N][N];

int sumA(array_t a) {
  int i, j;
  int sum = 0;
  for (i = 0; i < N; i++)
    for (j = 0; j < N; j++)
      sum += a[i][j];

  return sum;
}

int sumB(array_t a) {
  int i, j;
  int sum = 0;
  for (i = 0; i < N; i++)
    for (j = 0; j < N; j++)
      sum += a[j][i];

  return sum;
}

int sumC(array_t a) {
  int i, j;
  int sum = 0;
  for (i = 0; i < N; i+=2)
    for (j = 0; j < N; j+=2)
      sum += (a[j][i] + a[j][i+1] + a[j+1][i] + a[j+1][i+1])
}
```

C = 4096, B = 16, E = 1, S = 256

N = 64

sizeof(array_t) == 64 *64* == *4096* == *4*C

```
memory-cache graph

memory address start from 0 and end to 4096*4.

cell size is 16B. number(0-255) in cell means cache block number
 that the cell
will be cached.

    0    +---------+
         |    0    |
    16   +---------+
         |    1    |
    32   +---------+
         |    2    |
    48   +---------+
         |    .    |
         |    .    |
         |    .    |
         |    .    |
         |    .    |
4096-16  +---------+
         |   255   |
   4096  +---------+
         |    0    |
4096+16  +---------+
         |    1    |
4096+32  +---------+
         |    .    |
         |    .    |
         |    .    |
         |    .    |
         |    .    |
         |    .    |
4096*4-16+---------+
         |   255   |
4096*4   +---------+
```

A. sumA

```
sum += a[i][j];
```

read memory address order:

0, 4, 8, 12, ..., 4096*4-4

read cache order:

0, 0, 0, 0, 1, 1, 1, 1,.....255, 255, 255, 255, 0, 0, 0, 0,.....255, 255, 255, 255

first cache read miss, next 3 time read hit.

miss rate: 25%

B. sumB

```
sum += a[j][i];
```

read memory address order:

0, 644, 6442, ... 64463, 4, 644+4, 6442+4, ... 4096*4-4

read cache order:

0, 16, 32, 48, ... 240,(4 times) 1, 17, 33, ... 241,(4 times) 15, 31, 47, ... 255(4 times)

let's see first read loop:

read cache order loop 4 times

0, 16, 32, 48, ... 240,(4 times)

first loop all miss, next 3 loop all hit

so miss rate is 25%.

C. sumC

```
for (i = 0; i < N; i+=2)
  for (j = 0; j < N; j+=2)
    sum += (a[j][i] + a[j][i+1] + a[j+1][i] + a[j+1][i+1]);
```

easy to see that read a[j][i+1] and a[j+1][i+1] always hit

same like

```
for (i = 0; i < N; i+=2)
  for (j = 0; j < N; j+=2)
    sum += (a[j][i] + a[j+1][i]);
```

same like

```
for (i = 0; i < N; i+=2)
  for (j = 0; j < N; j++)
    sum += a[j][i];
```

total read count = 64*64

because of i+=2,

read cache order only loop 2 times

0, 16, 32, 48, ... 240,(2 times)

so miss rate is 50%

totol read miss count = 64/2 *64* 50% = 64*64/4

so miss rate is still 25%.

N = 60

A. sumA

```
sum += a[i][j];
```

read memory by step 1

miss rate 25%

B. sumB

```
for (i = 0; i < N; i++)
  for (j = 0; j < N; j++)
    sum += a[j][i];
```

it's interesting.

let's see first inner loop a[0][0] -> a[59][0]

read memory address order:

0, 604, 6042, ... 604*59

read cache order:

0, 15, 30, ..., 225, (17 numbers) 255, 14, 29, ..., 224, (17 numbers) 254, 13, 28, ..., 223, (17 numbers) 253, 12, 27, 42, 57, 72, 87, 102, 117 (9 numbers)

all read miss and store into different blocks

next 3 loops: a[0][1] -> a[59][1], a[0][2] -> a[59][2], a[0][3] -> a[59][3]

all hit

althrough N is smaller and not power of 2, miss rate is still 25%

C. sumC

same as miss rate when N = 64

25%

6.38

A.

4 * 16 * 16

B.

sizeof(point_color) == 16, B = 32

```
square[i][j].c = 0
```

miss, cache 2 point_color, then

```
square[i][j].m = 0
square[i][j].y = 0
square[i][j].k = 0
square[i][j+1].c = 0
square[i][j+1].m = 0
square[i][j+1].y = 0
square[i][j+1].k = 0
```

all hit

so miss count is 4 * 16 * 16 * 1/8

C.

1/8

6.39

A.

4 * 16 * 16

B.

sizeof(point_color) == 16, B = 32

```
square[j][i].c = 0
```

miss, cache 2 point_color, then

```
square[j][i].m = 0
square[j][i].y = 0
square[j][i].k = 0
```

all hit.

next loop

```
square[j+1][i] - square[j][i] == 16*16 == 256
```

square[j+1][i] miss, cache block not conflict with square[j][i]

```
square[j+8][i] - square[j][i] == 16*16*8 == 2048
```

square[j+8][i] miss, cache block overwrite square[j][i] block. so when we reach square[j][i+1], still miss.

so miss count is 4 * 16 * 16 * 1/4

C.

1/4

6.40

A.

4 *16* 16

B.

first loop, same like 6.38, but

write count is 16*16, miss rate is 1/2.

second loop, same like 6.39, but

write count is 16*16*3, miss rate is 1/6.

miss count is

16*16*1/2 + 16*16*3*1/6

C.

1/4

6.41

every loop

```
buffer[i][j].r = 0;
```

always miss, then cache one piexl, so

```
buffer[i][j].g = 0;
buffer[i][j].b = 0;
buffer[i][j].a = 0;
```

all hit

miss rate is 1/4

6.42

same like

```
for (i = 0; i < 640; i++)
  for (j = 0; j < 480; j++)
    buffer[i][j].r = 0;
    buffer[i][j].g = 0;
    buffer[i][j].b = 0;
    buffer[i][j].a = 0;
```

C = 64KB, B = 4B, sizeof(piexl) = 4

```
buffer[i][j].r = 0;
```

miss, cache one piexl, so

```
buffer[i][j].g = 0;
buffer[i][j].b = 0;
buffer[i][j].a = 0;
```

all hit

miss rate is 1/4

6.43

same like

```
for (i = 0; i < 640; i++)
  for (j = 0; j < 480; j++)
    (int*)&buffer[i][j] = 0;
```

every loop,

```
(int*)&buffer[i][j] = 0;
```

always miss

miss rate 100%

6.44

download mountain program from here

I've download it into chapter6/code dir

run

```
(cd chapter6/code/mountain; ./mountain)
```

see result

```
Clock frequency is approx. 2500.0 MHz
Memory mountain (MB/sec)
          s1        s2          s3          s4          s5          s
6         s7          s8          s9          s10         s11
 s12        s13         s14         s15
128m        12824       7552        5119        3776        2981
  2452        2080        1799        1663        1563        1483
    1410        1372        1334        1304
64m         12880       7575        5121        3790        2976
  2456        2079        1797        1663        1563        1489
    1415        1366        1333        1302
32m         12849       7635        5137        3785        2996
  2456        2075        1795        1671        1570        1486
    1419        1372        1321        1299
16m         12906       7656        5174        3826        3000
  2504        2090        1808        1700        1579        1506
    1442        1397        1373        1357
8m          13045       7832        5321        4072        3121
  2577        2194        1885        1787        1706        1663
    1633        1622        1621        1703
4m          13303       8352        5602        4210        3326
  2765        2352        2033        1931        1859        1824
    1812        1817        1906        2076
2m          16265       11003       9150        7368        6024
  5014        4310        3816        3795        3761        3848
    3877        3909        3946        4044
1024k       16708       11597       10180       8150        6605
  5544        4772        4169        4109        4150        4136
```

```
    4127       4113       4127       4119
512k       16674      11613      10179      8160       6595
  5543       4787       4182       4245       4242       4300
    4360       4416       4508       4668
256k       16929      12872      11826      10661      8538
  7360       6383       5724       6516       6951       7181
    7297       7041       7634       7768
128k       16992      15381      14141      13145      12497
  11444      10028      9734       9642       9638       9671
    9696       9606       9438       9117
64k        17109      15238      14163      13563      12263
  11302      9736       9412       9175       9162       9239
    9200       9267       9229       12245
32k        18060      17655      17503      16954      16785
  16489      16718      14382      16135      16512      16471
    15799      14318      16250      15337
16k        17809      17534      17326      16516      16783
  16406      16250      13333      14043      15280      13890
    12546      13347      12394      17062
```

6.45

assume matrix size N = 4, cache block size B = 8Byte, sizeof(int) = 4

```
matrix size 4*4
+--+--+--+--+
|0 |1 |2 |3 |
+--+--+--+--+
|4 |5 |6 |7 |
+--+--+--+--+
|8 |9 |10|11|
+--+--+--+--+
|12|13|14|15|
+--+--+--+--+
```

function transponse

```
void transpose(int* dst, int* src, int N) {
  int i, j;

  for (i = 0; i <= N-1; i++)
    for (j = 0; j <= N-1; j++)
      dst[j*N+i] = src[i*N+j];
}
```

every cache block can store 2 int numbers

traverse src by step 1, order:

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

order 0: cache miss, load src[0],src[1] order 1: cache hit because of order 0 order 2: cache miss, load src[2],src[3] .... order 15: cache hit

if B is greater, hit rate will be greater too.

traverse dst by step 4, order:

0 4 8 12 1 5 9 13 2 6 10 14 3 7 11 15

element 0: miss, load dst[0],dst[1] element 4: miss, load dst[4],dst[5] element 8: miss, load dst[8],dst[9] element 12: miss, load dst[12],dst[13] element 1: interesting, order 0 has loaded dst[1] into cache, hit. but in many cases, element 4/8/12 may use the same cache block with element 0. so miss is also possible. .... element 15: hit / miss

let's assume worest case: all element miss cache

if we split matrix by 2*2

```
split matrix by size 2*2 block
+--+--+--+--+
|     |     |
+  0  +  1  +
|     |     |
+--+--+--+--+
|     |     |
+  2  +  3  +
|     |     |
+--+--+--+--+
```

transpose block 0 itself

transpose block 1 with 2

transpose block 3 itself

code

```
  for (i = 0; i <= N-2; i+=2)
    for (j = 0; j <= N-2; j+=2) {
      dst[j*N+i] = src[i*N+j];
      dst[j*N+i+1] = src[(i+1)*N+j];
      dst[(j+1)*N+i] = src[i*N+j+1];
      dst[(j+1)*N+i+1] = src[(i+1)*N+j+1];
    }
```

when i = 0, j = 0, transpose block 0 itself

```
+--+--+      +--+--+
|0 |1 |      |0 |4 |
+--+--+  =>  +--+--+
|4 |5 |      |1 |5 |
+--+--+      +--+--+

dst[0] = src[0];
dst[1] = src[4];
dst[4] = src[1];
dst[5] = src[5];
```

if element 0 is miss, element 1 must hit; if element 4 is miss, element 5 must hit;

50% is the highest hit rate in such low cache block size.

if B is greater and cache size C is larger, we can split matrix into 4*4, 8*8 or more larger. theoretically we will archive the highest hit rate.

finally code:

```
/*
 * transpose.c
 */
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>

#define LOOP 1000
#define MAX  1024
#define LEN  MAX*MAX
#define BLOCK 16

void transpose(int* dst, int* src, int N) {
  int i, j;

  for (i = 0; i <= N-1; i++)
    for (j = 0; j <= N-1; j++)
      dst[j*N+i] = src[i*N+j];
}
```

```
void effective_transpose(int* dst, int* src, int N) {
  int i, j, a, b;

  for (i = 0; i <= N-BLOCK; i+=BLOCK)
    for (j = 0; j <= N-BLOCK; j+=BLOCK)
      for (a = i; a < i+BLOCK; a++)
        for (b = j; b < j+BLOCK; b++)
          dst[b*N+a] = src[a*N+b];

  for (; i <= N-1; i++)
    for (; j <= N-1; j++)
      dst[j*N+i] = src[i*N+j];
}

void test(void) {
  int* d = (int*)malloc(sizeof(int)*LEN);
  int* s = (int*)malloc(sizeof(int)*LEN);

  transpose(d, s, MAX);

  for (int i = 0; i < MAX; i++)
    for (int j = 0; j < MAX; j++)
      assert(s[i*MAX+j] == d[j*MAX+i]);

  effective_transpose(d, s, MAX);

  for (int i = 0; i < MAX; i++)
    for (int j = 0; j < MAX; j++)
      assert(s[i*MAX+j] == d[j*MAX+i]);

  free((void*)d);
  free((void*)s);
}

void prof(void) {
  int* d = (int*)malloc(sizeof(int)*LEN);
  int* s = (int*)malloc(sizeof(int)*LEN);

  for (int c = 0; c < LOOP; c++)
```

```
    transpose(d, s, MAX);

  free((void*)d);
  free((void*)s);
}

void prof_effect(void) {
  int* d = (int*)malloc(sizeof(int)*LEN);
  int* s = (int*)malloc(sizeof(int)*LEN);

  for (int c = 0; c < LOOP; c++)
    effective_transpose(d, s, MAX);

  free((void*)d);
  free((void*)s);
}

int main(int argc, char* argv[]) {
  /*test();*/

  /*prof();*/

  prof_effect();

  return 0;
}
```

in code, matrix size 1024*1024, loop 1000 times to measure program run time.

change BLOCK from 2 to 16, record time statistics

origin function run time: 16.46s

BLOCK	time(s)
2	9.99
3	7.16
4	5.6
5	5.66
6	5.34
7	5.39
8	5.38
9	5.48
10	6.21
11	7.9
12	10.17
13	11.14
14	11.88
15	12.11
16	11.85

> tip: look up cpu cache info
>
> cat /sys/devices/system/cpu/cpu0/cache/*

6.46

same like 6.45, pay attention to brilliant comment :)

```
/*
 * convert.c
 */
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>

#define LOOP 1000
#define MAX  1024
#define LEN  MAX*MAX
#define BLOCK 16

void convert(int* src, int N) {
  int i, j;

  for (i = 0; i <= N-1; i++)
    for (j = 0; j <= N-1; j++)
      src[j*N+i] = src[i*N+j] || src[j*N+i];
}

void effective_convert(int* src, int N) {
  int i, j, a, b, tmp;

  for (i = 0; i <= N-BLOCK; i+=BLOCK)
    /* brilliant! not j = 0 here */
    for (j = i; j <= N-BLOCK; j+=BLOCK)
      for (a = i; a < i+BLOCK; a++)
        for (b = j; b < j+BLOCK; b++) {
          /* brilliant! store two value in one loop */
          tmp = src[b*N+a] || src[a*N+b];
          src[b*N+a] = tmp;
          src[a*N+b] = tmp;
        }

  for (; i <= N-1; i++)
    for (; j <= N-1; j++)
```

```
      src[j*N+i] = src[i*N+j] || src[j*N+i];
}

void test(void) {
  int* s = (int*)malloc(sizeof(int)*LEN);
  int* e = (int*)malloc(sizeof(int)*LEN);

  convert(s, MAX);
  effective_convert(e, MAX);

  for (int i = 0; i < MAX; i++)
    for (int j = 0; j < MAX; j++)
      assert(s[i*MAX+j] == e[i*MAX+j]);

  free((void*)s);
  free((void*)e);
}

void prof(void) {
  int* s = (int*)malloc(sizeof(int)*LEN);

  for (int c = 0; c < LOOP; c++)
    convert(s, MAX);

  free((void*)s);
}

void prof_effect(void) {
  int* s = (int*)malloc(sizeof(int)*LEN);

  for (int c = 0; c < LOOP; c++)
    effective_convert(s, MAX);

  free((void*)s);
}

int main(int argc, char* argv[]) {
  /*test();*/

  /*prof();*/
```

```
  prof_effect();

  return 0;
}
```

measure time statistics:

origin function run time: 30.28s

BLOCK	time(s)
1	14.26
2	12.01
3	7.43
4	6.20
5	6.08
6	5.86
7	5.70
8	5.67
9	6.30
10	6.39
11	6.21
12	6.18
13	5.9
14	6.3
15	5.88
16	5.92

Linking

> Our ultimate goal is extensible programming. By this, we mean the construction of hierarchies of modules, each module adding new functionality to the system.
>
> by Niklaus Wirth

7.1 - 7.5 visit book

7.6 - 7.13 visit here

test

code directory: `./code`

test way:

- assert means assert function from `<assert.h>`
- output means to watch code output to judge if it works right

solution	code file	test way
7.6	7.6/swap.c	----
7.7	7.7/*	output
7.8	----	----
7.9	7.9/*	output
7.10	----	----
7.11	----	----
7.12	----	----
7.13	7.13/*	----

7.6

```
/*
 * swap.c
 */

extern int buf[];

int* bufp0 = &buf[0];
static int* bufp1;

static void incr() {
  static int count=0;
  count++;
}

void swap() {
  int temp;

  incr();
  bufp1 = &buf[1];
  temp = *bufp0;
  *bufp0 = *bufp1;
  *bufp1 = temp;
}
```

	in .symtab?	type	module	section
buf	Yes	external	m	.data
bufp0	Yes	global	swap	.data
bufp1	Yes	local	swap	.bss
swap	Yes	global	swap	.text
temp	No	------	-----	------
incr	Yes	local	swap	.text
count	Yes	local	swap	.bss

```
(cd chapter7/code/7.6; make && make sym)
```

output:

```
gcc -c swap.c
objdump -t swap.o

swap.o：     文件格式 elf64-x86-64

SYMBOL TABLE:
0000000000000000 l     O .bss   0000000000000008 bufp1
0000000000000000 l     F .text    0000000000000015 incr
0000000000000008 l     O .bss   0000000000000004 count.1747
0000000000000000 g     O .data    0000000000000008 bufp0
0000000000000000         *UND*    0000000000000000 buf
0000000000000015 g     F .text    0000000000000049 swap
```

7.7

```
/*
 * bar5.c
 */
double x;

void f() {
  /*x = -0.0;*/
}
```

delete line

```
x = -0.0;
```

7.8

A.

main.1

main.2

B.

unknown

unknown

C.

error

error

7.9

function main print `0x55` on my machine.

modify `char main` to `unsigned int main` here

```
/*
 * bar6.c
 */
#include <stdio.h>

unsigned int main;

void p2() {
  printf("0x%x\n", main);
}
```

```
/*
 * foo6.c
 */
void p2(void);

void offset(void) {
  return;
}

int main(int argc, char* argv[]) {
  p2();
  return 0;
}
```

```
(cd chapter7/code/7.9; make && ./main)
```

output:

```
gcc foo6.c bar6.c -o main
/usr/lib/gcc/x86_64-pc-linux-gnu/4.9.4/../../../../x86_64-pc-lin
ux-gnu/bin/ld: Warning: alignment 1 of symbol `main' in /tmp/ccT
BhRjm.o is smaller than 4 in /tmp/ccc3SjbF.o

0xe5894855
```

using objdump inspect

```
objdump -d main
```

find function main

```
000000000040055d <main>:
  40055d:    55                      push   %rbp
  40055e:    48 89 e5                mov    %rsp,%rbp
  400561:    48 83 ec 10             sub    $0x10,%rsp
  400565:    89 7d fc                mov    %edi,-0x4(%rbp)
  400568:    48 89 75 f0             mov    %rsi,-0x10(%rbp)
  40056c:    e8 07 00 00 00          callq  400578 <p2>
  400571:    b8 00 00 00 00          mov    $0x0,%eax
  400576:    c9                      leaveq
  400577:    c3                      retq
```

`0xe5894855` is first 2 instructions content of function main.

works same like

```
/*
 * another-bar6.c
 */
#include <stdio.h>

int main(int argc, char* argv[]);

void p2() {
  printf("0x%x\n", * (unsigned int *)main);
}
```

7.10

A.

```
gcc p.o libx.a
```

B.

```
gcc p.o libx.a liby.a libx.a
```

C.

```
gcc p.o libx.a liby.a libx.a libz.a
```

7.11

space for section .bss

7.12

A.

ADDR(s) = ADDR(.text) = 0x4004e0

ADDR(r.symbol) = ADDR(swap) = 0x4004f8

refaddr = ADDR(s) + r.offset = 0x4004ea

*refptr = (unsigned) (ADDR(r.symbol) + r.addend - refaddr) = 0xa

B.

ADDR(s) = ADDR(.text) = 0x4004d0

ADDR(r.symbol) = ADDR(swap) = 0x400500

refaddr = ADDR(s) + r.offset = 0x4004da

*refptr = (unsigned) (ADDR(r.symbol) + r.addend - refaddr) = 0x22

7.13

A.

libm.a path

```
whereis libm.a
```

output:

```
libm: /usr/lib64/libm.a /usr/lib64/libm.so
```

libm.a files

```
ar t /usr/lib64/libm.a
```

output:

```
s_lib_version.o
s_matherr.o
s_signgam.o
fclrexcpt.o
fgetexcptflg.o
fraiseexcpt.o
fsetexcptflg.o
ftestexcept.o
fegetround.o
fesetround.o
fegetenv.o
feholdexcpt.o
fesetenv.o
feupdateenv.o
t_exp.o
fedisblxcpt.o
feenablxcpt.o
fegetexcept.o
powl_helper.o
e_acos.o
e_acosh.o
e_asin.o
e_atan2.o
e_atanh.o
....
```

similar way for libc.a

B.

compile code with -Og and -Og -g

```c
/*
 * little.c
 */
int main(int argc, char* argv[]) {
  return 0;
}
```

```
(cd chapter7/code/7.13; make && make dump-exe-code)
# objdump -d og-little
# objdump -d dog-little
```

they are same

C.

```
ldd og-little
```

output:

```
linux-vdso.so.1 (0x00007ffef51d3000)
libc.so.6 => /lib64/libc.so.6 (0x00007f27c6b8b000)
/lib64/ld-linux-x86-64.so.2 (0x00007f27c6f24000)
```

Exceptional Control Flow

> Nature provides exceptions to every rule.
>
> by Margaret Fuller

`csapp.h & csapp.c` is downloaded from csapp site

test

8.1 - 8.8 visit book

8.9 - 8.26 visit here

test

code directory: `./code`

test way:

- output means to watch code output to judge if it works right

solution	code file	test way
8.9	----	----
8.10	----	----
8.11	8.11.c	output
8.12	8.12.c	output
8.13	8.13.c	output
8.14	8.14.c	output
8.15	8.15.c	output
8.16	8.16.c	output
8.17	----	----
8.18	8.18.c	output
8.19	8.19.c	output
8.20	8.20.c	output
8.21	8.21.c	output
8.22	mysystem.c, exit-code.c, wait-sig.c	output
8.23	8.23.c	output
8.24	8.24.c	output
8.25	8.25.c	output
8.26	shell/*	output

8.9

process pair	cocurrent?
AB	No
AC	Yes
AD	Yes
BC	Yes
BD	Yes
CD	Yes

8.10

A. call once, return twice

fork

B. call once, never return

longjmp, execve

C. call once, return 1 or more times

setjmp

8.11

4 lines

```
                    c
                +-------+
                |    "hello"
                |
            c   |   p
        +-------+-------+
        |     fork   "hello"
        |
        |           c
        |       +-------+
        |       |    "hello"
        |       |
        |   p   |   p
 +------+-------+-------+
main  fork    fork   "hello"
```

8.12

8 lines

```
                    c
                +-------+---------+
                |    "hello"   "hello"
                |
            c   |   p
        +-------+-------+---------+
        |     fork   "hello"   "hello"
        |
        |           c
        |       +-------+---------+
        |       |    "hello"   "hello"
        |       |
        |   p   |   p
 +------+-------+-------+---------+
main  fork    fork   "hello"   "hello"
```

8.13

```
x=4
x=3
x=2
```

pay attention, parent process and child process don't share global x, they have own private x.

8.14

3 lines

```
                         c
                     +-------+
                     |    "hello"
                     |
                 c   |   p
             +-------+-------+
             |     fork   "hello"
             |
             |   p
      +------+-------+-------+
     main  fork   return  "hello"
```

8.15

5 lines

```
                    c
                +-------+---------+
                |    "hello"   "hello"
                |
            c   |   p
        +-------+-------+---------+
        |     fork   "hello"   "hello"
        |
        |
        |   p
 +------+-------+-------+
main  fork   return  "hello"
```

8.16

```
counter = 2
```

child process has its own counter.

8.17

```
hello   0  1  Bye   2   Bye
hello   1  0  Bye   2   Bye
hello   1  Bye  0   2   Bye
```

8.18

```
                    c
                +-------+---------+
                |       "0"     exit "2"
                |
            c   |   p
        +-------+-------+---------+
        |     fork      "1"     exit "2"
        |   (atexit)
        |           c
        |       +-------+---------+
        |       |       "0"      exit
        |       |
        |   p   |   p
 +------+-------+-------+---------+
main  fork    fork      "1"      exit
```

2 must be behind 0/1

B & D is impossible.

8.19

2^n

8.20

```
/*
 * 8.20.c
 */
#include <stdio.h>
#include "csapp.h"

int main(int argc, char* argv[], char* env[]) {
  if (execve("/bin/ls", argv, env) == -1) {
    fprintf(stderr, "execve error: %s\n", strerror(errno));
    exit(1);
  }
}
```

8.21

```
abc
```

or

```
bac
```

8.22

```
/*
 * mysystem.c
 */
#include <stdio.h>
#include "csapp.h"

int mysystem(char* command) {
  pid_t pid;
  int status;

  if ((pid = Fork()) == 0) {
    /* child process */
    char* argv[4] = { "", "-c", command, NULL };
    execve("/bin/sh", argv, environ);
  }

  /* print child pid so we can kill it */
  printf("child pid: %d\n", pid);

  if (Waitpid(pid, &status, 0) > 0) {
    /* exit normally */
    if (WIFEXITED(status))
      return WEXITSTATUS(status);

    /* exit by signal */
    if (WIFSIGNALED(status))
      return WTERMSIG(status);
  }
}

int main(int argc, char* argv[]) {
  int code;

  code = mysystem("./exit-code");
  printf("normally exit, code: %d\n", code); fflush(stdout);

  code = mysystem("./wait-sig");
  printf("exit caused by signal, code: %d\n", code); fflush(stdo
```

```
ut);
  return 0;
}
```

when running `./mysystem`, it runs `./exit-code`

```
/*
 * exit-code.c
 */
#include "csapp.h"

int main(int argc, char* argv[]) {
  exit(10);
}
```

should output

```
normally exit, code 10
```

and runs `./wait-sig`, stuck here

```
/*
 * wait-sig.c
 */
#include "csapp.h"

int main(int argc, char* argv[]) {
  while (1);
}
```

fall into dead loop. open another terminal, type

```
kill -<n> <wait-sig's pid>
```

`./mysystem` will return and output

```
exit caused by signal, code <n>
```

8.23

```
    SIGUSR2       SIGUSR2  SIGUSR2  SIGUSR2  SIGUSR2
       |             |        |        |        |
being handled   Pending  Canceld  Canceld  Canceld
 need 1 sec
```

there's only one pending signal all the time. other same type signals will be canceled.

if you modify code

remove sleep

```
/* sleep(1); */
```

or

send more signals

```
for (i = 0; i < 500000; i++)
```

you will get different counter output.

8.24

```
/* $begin waitpid1 */
#include "csapp.h"
#define N 2
#define LEN 100

int main() {
  int status, i;
  pid_t pid;

  /* Parent creates N children */
  for (i = 0; i < N; i++)
    if ((pid = Fork()) == 0) {
      /* access address 0, cause fault */
      char* cptr = NULL;
      *cptr = 'd';
    }

  /* Parent reaps N children in no particular order */
  while ((pid = waitpid(-1, &status, 0)) > 0) {
    if (WIFEXITED(status))
      printf("child %d terminated normally with exit status=%d\n"
,
          pid, WEXITSTATUS(status));
    else if (WIFSIGNALED(status)) {
      /* print signal that cause process exit */
      char buf[LEN];
      sprintf(buf, "child %d terminated by signal %d", pid, WTER
MSIG(status));
      psignal(WTERMSIG(status), buf);
    }
    else
      printf("child %d terminated abnormally\n", pid);
  }

  /* The only normal termination is if there are no more childre
n */
  if (errno != ECHILD)
    unix_error("waitpid error");
```

```
  exit(0);
}
/* $end waitpid1 */
```

8.25

```c
/*
 * 8.25.c
 */
#include <stdio.h>
#include "csapp.h"

sigjmp_buf buf;

void handler(int sig) {
  /* jump */
  siglongjmp(buf, 1);
}

char* tfgets(char* s, int size, FILE* stream) {
  char* result;

  if (!sigsetjmp(buf, 1)) {
    alarm(5);
    if (signal(SIGALRM, handler) == SIG_ERR)
      unix_error("set alarm handler error");
    return fgets(s, size, stream);
  } else {
    /* run out of time */
    return NULL;
  }
}

#define LEN 100

int main(int argc, char* argv[]) {
  char buf[LEN];
  char* input = tfgets(buf, LEN, stdin);

  if (input == NULL) {
    printf("nothing input: NULL\n");
  } else {
    printf("%s", input);
  }
```

```
    return 0;
}
```

8.26

learn

- origin shell code in pic 8-23, pic 8-24, pic 8-25
- how to use waitpid in pic 8-18
- how to wait child process in pic 8-42

and write a job management module

compose them and the little shell is born.

add `bg` `fg` `jobs` command to origin shell program

```
#include <assert.h>
#include "../csapp.h"
#include "shell.h"
#include "job.h"

void eval(char *cmdline)
{
  char *argv[MAXARGS]; /* Argument list execve() */
  char buf[MAXLINE];   /* Holds modified command line */
  int bg;              /* Should the job run in bg or fg? */
  pid_t pid;           /* Process id */

  strcpy(buf, cmdline);
  bg = parse_line(buf, argv);
  if (argv[0] == NULL)
    return;   /* Ignore empty lines */

  if (!builtin_command(argv)) {
    sigset_t mask_one, prev_one;
    Sigemptyset(&mask_one);
    Sigaddset(&mask_one, SIGCHLD);

    /* block signal child */
    Sigprocmask(SIG_BLOCK, &mask_one, &prev_one);
    if ((pid = Fork()) == 0) {
      /* unblock in child process */
      Sigprocmask(SIG_SETMASK, &prev_one, NULL);
```

```
      /* set gid same like pid */
      Setpgid(0, 0);

      if (execve(argv[0], argv, environ) < 0) {
        printf("%s: Command not found.\n", argv[0]);
        exit(0);
      }
    }

    sigset_t mask_all, prev_all;
    Sigfillset(&mask_all);
    // save job info
    Sigprocmask(SIG_BLOCK, &mask_all, &prev_all);
    Jid new_jid = new_job(pid, cmdline, !bg);
    Sigprocmask(SIG_SETMASK, &prev_all, NULL);

    if (!bg) {
      set_fg_pid(pid);
      while(get_fg_pid())
        sigsuspend(&prev_one);
    }
    else
      printf("[%d] %d %s \t %s\n", new_jid, pid, "Running", cmdl
ine);

    /* unblock child signal */
    Sigprocmask(SIG_SETMASK, &prev_one, NULL);
  }
  return;
}

/*
 * If first arg is a builtin command, run it and return true;
 * else return false.
 */
int builtin_command(char **argv)
{
  if (!strcmp(argv[0], "quit")) /* quit command */
    exit(0);
```

```
  if (!strcmp(argv[0], "&"))    /* Ignore singleton & */
    return 1;
  if (!strcmp(argv[0], "jobs")) {
    print_jobs();
    return 1;
  }
  // > fg
  if (!strcmp(argv[0], "fg")) {
    int id;
    // right format: fg %ddd or fg ddd
    if ((id = parse_id(argv[1])) != -1 && argv[2] == NULL) {
      sigset_t mask_one, prev_one;
      Sigemptyset(&mask_one);
      Sigaddset(&mask_one, SIGCHLD);
      Sigprocmask(SIG_BLOCK, &mask_one, &prev_one);

      pid_t pid = id;
      // if param is jid
      if (argv[1][0] == '%') {
        JobPtr jp = find_job_by_jid(id);
        pid = jp->pid;
      }
      Kill(pid, SIGCONT);
      set_fg_pid(pid);
      while(get_fg_pid())
        sigsuspend(&prev_one);

      Sigprocmask(SIG_SETMASK, &prev_one, NULL);

    } else {
      printf("format error, e.g. fg %%12 || fg 1498\n");
    }

    return 1;
  }
  // > bg
  if (!strcmp(argv[0], "bg")) {
    int id;
    // right format: bg %ddd or bg ddd
    if ((id = parse_id(argv[1])) != -1 && argv[2] == NULL) {
```

```
      pid_t pid = id;
      // jid param
      if (argv[1][0] == '%') {
        JobPtr jp = find_job_by_jid(id);
        pid = jp->pid;
      }
      Kill(pid, SIGCONT);
    } else {
      printf("format error, e.g. bg %%12  or  bg 1498\n");
    }
    return 1;
  }

  return 0;                     /* Not a builtin command */
}

/* parse_line - Parse the command line and build the argv array
*/
int parse_line(char *buf, char **argv)
{
  char *delim;         /* Points to first space delimiter */
  int argc;            /* Number of args */
  int bg;              /* Background job? */

  buf[strlen(buf)-1] = ' ';  /* Replace trailing '\n' with space
 */
  while (*buf && (*buf == ' ')) /* Ignore leading spaces */
    buf++;

  /* Build the argv list */
  argc = 0;
  while ((delim = strchr(buf, ' '))) {
    argv[argc++] = buf;
    *delim = '\0';
    buf = delim + 1;
    while (*buf && (*buf == ' ')) /* Ignore spaces */
      buf++;
  }
  argv[argc] = NULL;
```

```
  if (argc == 0)  /* Ignore blank line */
    return 1;

  /* Should the job run in the background? */
  if ((bg = (*argv[argc-1] == '&')) != 0)
    argv[--argc] = NULL;

  return bg;
}

static int is_number_str(char* s) {
  int len = strlen(s);
  for (int i = 0; i < len; i++)
    if (!isdigit(s[i]))
      return 0;

  return 1;
}

int parse_id(char* s) {
  int error = -1;
  if (s == NULL)
    return error;

  /* format: %ddddd */
  if (s[0] == '%') {
    if (!is_number_str(s+1))
      return error;

    return atoi(s+1);
  }
  /* format: dddddd */
  if (is_number_str(s))
    return atoi(s);

  /* not right */
  return error;
}

void test_shell() {
```

```
  // parse id
  assert(-1 == parse_id("ns"));
  assert(-1 == parse_id("%%"));
  assert(0 == parse_id("%0"));
  assert(0 == parse_id("0"));
  assert(98 == parse_id("%98"));
  assert(98 == parse_id("98"));
}
```

job management module. signal handlers are the key part

```
/*
 * job.c
 */
#include <stdio.h>
#include <assert.h>
#include "job.h"
#include "../csapp.h"

static volatile sig_atomic_t fg_pid;
static Job jobs[MAXJOBS];

int is_fg_pid(pid_t pid) {
  return fg_pid == pid;
}
pid_t get_fg_pid() {
  return fg_pid;
}
void set_fg_pid(pid_t pid) {
  fg_pid = pid;
}

/* SIGCONT signal */
void sigchild_handler(int sig) {
  int old_errno = errno;
  int status;
  pid_t pid;

  sigset_t mask_all, prev_all;
```

```
  Sigfillset(&mask_all);

  /* exit or be stopped or continue */
  while ((pid = waitpid(-1, &status, WNOHANG|WUNTRACED|WCONTINUE
D)) > 0) {
    /* exit normally */
    if (WIFEXITED(status) || WIFSIGNALED(status)) {
      if (is_fg_pid(pid)) {
        set_fg_pid(0);
      } else {
        Sio_puts("pid "); Sio_putl(pid); Sio_puts(" terminates\n"
);
      }
      Sigprocmask(SIG_BLOCK, &mask_all, &prev_all);
      del_job_by_pid(pid);
      Sigprocmask(SIG_SETMASK, &prev_all, NULL);
    }

    /* be stopped */
    if (WIFSTOPPED(status)) {
      if (is_fg_pid(pid)) {
        set_fg_pid(0);
      }
      // set pid status stopped
      Sigprocmask(SIG_BLOCK, &mask_all, &prev_all);
      JobPtr jp = find_job_by_pid(pid);
      set_job_status(jp, Stopped);
      Sigprocmask(SIG_SETMASK, &prev_all, NULL);

      Sio_puts("pid "); Sio_putl(pid); Sio_puts(" be stopped\n")
;
    }

    /* continue */
    if(WIFCONTINUED(status)) {
      set_fg_pid(pid);
      // set pid status running
      Sigprocmask(SIG_BLOCK, &mask_all, &prev_all);
      JobPtr jp = find_job_by_pid(pid);
      set_job_status(jp, Running);
```

```
        Sigprocmask(SIG_SETMASK, &prev_all, NULL);

        Sio_puts("pid "); Sio_putl(pid); Sio_puts(" continue\n");
      }
    }

    errno = old_errno;
  }

  void sigint_handler(int sig) {
    /* when fg_pid == 0, stop shell itself, it'll be a dead loop */

    if (is_fg_pid(0)) {
      Signal(SIGINT, SIG_DFL);
      Kill(getpid(), SIGINT);
    } else {
      Kill(get_fg_pid(), SIGINT);
    }
  }

  void sigstop_handler(int sig) {
    /* same like int handler */
    if (is_fg_pid(0)) {
      Signal(SIGTSTP, SIG_DFL);
      Kill(getpid(), SIGTSTP);
    } else {
      Kill(get_fg_pid(), SIGTSTP);
    }
  }

  JobPtr find_job_by_jid(Jid jid) {
    return &(jobs[jid]);
  }

  JobPtr find_job_by_pid(pid_t pid) {
    for (int i = 0; i < MAXJOBS; i++) {
      Job j = jobs[i];
      if (j.using && j.pid == pid) {
        return &(jobs[i]);
      }
```

```
  }
  /* no such job */
  return NULL;
}

void set_job_status(JobPtr jp, enum JobStatus status) {
  if (jp)
    jp->status = status;
}


// seek a spare place for new job
static int find_spare_jid() {
  Jid jid = -1;
  for (int i = 0; i < MAXJOBS; i++) {
    if (jobs[i].using == 0) {
      jid = i;
      break;
    }
  }
  return jid;
}

int new_job(pid_t pid, char* cmdline, int fg) {
  // find a jid
  Jid jid = find_spare_jid();
  if (jid == -1)
    unix_error("no more jid to use");

  // save process info
  jobs[jid].jid = jid;
  jobs[jid].pid = pid;
  jobs[jid].status = Running;
  strcpy(jobs[jid].cmdline, cmdline);
  jobs[jid].using = 1;

  return jid;
}

void del_job_by_pid(pid_t pid) {
```

```
  // search job whose pid is pid
  for (int i = 0; i < MAXJOBS; i++) {
    if (jobs[i].using && jobs[i].pid == pid) {
      // delete job
      jobs[i].using = 0;
    }
  }
}

void print_jobs() {
  for (int i = 0; i < MAXJOBS; i++) {
    Job j = jobs[i];
    if (j.using) {
      printf("[%d] %d %s \t %s\n", j.jid, j.pid,
          j.status == Running ? "Running" : "Stopped", j.cmdline
);
    }
  }
}

void init_jobs() {
  memset(jobs, 0, sizeof(jobs));
}

void test_job() {

}
```

test it

```
(cd chapter8/code/shell; make && ./shell)
```

ps: `./loop` is a dead loop program, `./sleep` sleeps 5 secs and exit.

```
> jobs
> ./loop
^Zpid 4948 be stopped
> jobs
[0] 4948 Stopped         ./loop

> fg %0
pid 4948 continue
^Zpid 4948 be stopped
> ./loop &
[1] 4950 Running         ./loop &

> jobs
[0] 4948 Stopped         ./loop

[1] 4950 Running         ./loop &

> bg 4948
> pid 4948 continue

> jobs
[0] 4948 Running         ./loop

[1] 4950 Running         ./loop &

> quit
```

Virtual Memory

> I wanted to have virtual memory, at least as it's coupled with file systems.
>
> by Ken Thompson

9.1 - 9.10 visit book

9.11 - 9.20 visit here

test

code directory: `./code`

test way:

- assert means assert function from `<assert.h>`
- output means to watch code output to judge if it works right

solution	code file	test way
9.11	----	----
9.12	----	----
9.13	----	----
9.14	9.14.c	assert
9.15	----	----
9.16	----	----
9.17	vm/(mm.9.17, 9.17.c)	assert
9.18	vm/(mm.9.18, 9.18.c)	assert
9.19	----	----
9.20	malloc/*	measure performance with std malloc

9.11

VA: 0x027c

A.

```
 13 12 11 10  9  8  7  6  5  4  3  2  1  0
+--|--|--|--|--|--|--|--|--|--|--|--|--|--+
| 0| 0| 0| 0| 1| 0| 0| 1| 1| 1| 1| 1| 0| 0|
+--|--|--|--|--|--|--|--|--|--|--|--|--|--+
```

B.

param	value
VPN	0x09
TLBI	0x01
TLBT	0x02
hit?	No
page falut?	No
PPN	0x17

C.

```
 11 10  9  8  7  6  5  4  3  2  1  0
+--|--|--|--|--|--|--|--|--|--|--|--+
| 0| 1| 0| 1| 1| 1| 1| 1| 1| 1| 0| 0|
+--|--|--|--|--|--|--|--|--|--|--|--+
```

D.

param	value
CO	0x00
CI	0x0F
CT	0x17
hit?	No
value	-----

9.12

VA: 0x03a9

A.

```
 13 12 11 10  9  8  7  6  5  4  3  2  1  0
+--|--|--|--|--|--|--|--|--|--|--|--|--|--+
| 0| 0| 0| 0| 1| 1| 1| 0| 1| 0| 1| 0| 0| 1|
+--|--|--|--|--|--|--|--|--|--|--|--|--|--+
```

B.

param	value
VPN	0x0E
TLBI	0x02
TLBT	0x03
hit?	No
page falut?	No
PPN	0x11

C.

```
 11 10  9  8  7  6  5  4  3  2  1  0
+--|--|--|--|--|--|--|--|--|--|--|--+
| 0| 1| 0| 0| 0| 1| 1| 0| 1| 0| 0| 1|
+--|--|--|--|--|--|--|--|--|--|--|--+
```

D.

param	value
CO	0x01
CI	0x0A
CT	0x11
hit?	No
value	-----

9.13

VA: 0x0040

A.

```
  13 12 11 10  9  8  7  6  5  4  3  2  1  0
+--|--|--|--|--|--|--|--|--|--|--|--|--|--+
| 0| 0| 0| 0| 0| 0| 0| 1| 0| 0| 0| 0| 0| 0|
+--|--|--|--|--|--|--|--|--|--|--|--|--|--+
```

B.

param	value
VPN	0x01
TLBI	0x01
TLBT	0x00
hit?	No
page falut?	Yes

9.14

```c
/*
 * 9.14.c
 */
#include <stdio.h>
#include <assert.h>
#include "vm/csapp.h"

void test(char* filename, char* content) {
  int fd;
  char buf[20];
  fd = Open(filename, O_RDONLY, 0);
  Read(fd, buf, strlen(content));
  assert( !strncmp(buf, content, strlen(content)) );
}

int touch(char* filename, char* content) {
  int fd;
  umask(DEF_UMASK);
  fd = Open(filename, O_WRONLY|O_CREAT|O_TRUNC, DEF_MODE);
  Write(fd, content, strlen(content));
  Close(fd);
}

int main(int argc, char* argv[]) {
  touch("hello.txt", "Hello, world!");
  test("hello.txt", "Hello, world!");

  struct stat stat;
  int fd;
  char* bufp;
  size_t size;

  fd = Open("hello.txt", O_RDWR, 0);
  fstat(fd, &stat);
  size = stat.st_size;

  bufp = Mmap(NULL, size, PROT_WRITE, MAP_SHARED, fd, 0);
  *bufp = 'J';
```

```
    Munmap(bufp, size);

    test("hello.txt", "Jello, world!");
    return 0;
}
```

9.15

malloc	size	header
malloc(3)	8	0x9
malloc(11)	16	0x11
malloc(20)	24	0x19
malloc(21)	24	0x19

9.16

alignment	allocated block	spare block	min block size
word	Header & Footer	Header & Footer	16
word	Header	Header & Footer	16
double word	Header & Footer	Header & Footer	16
double word	Header	Header & Footer	16

9.17

complete code in `chapter9/code/vm/mm.9.17.c`

```
--- mm.c    2017-11-09 02:57:43.679935907 +0000
+++ mm.9.17.c    2017-11-09 02:57:43.679935907 +0000
@@ -41,6 +41,7 @@

 /* Global variables */
 static char *heap_listp = 0;  /* Pointer to first block */
+static char *rover;           /* Next fit rover */

 /* Function prototypes for internal helper routines */
 static void *extend_heap(size_t words);
@@ -69,6 +70,7 @@
   heap_listp += (2*WSIZE);                     //line:vm:mm:end
init
   /* $end mminit */

+  rover = heap_listp;
   /* $begin mminit */

   /* Extend the empty heap with a free block of CHUNKSIZE bytes
 */
@@ -177,6 +179,10 @@
     bp = PREV_BLKP(bp);
   }
   /* $end mmfree */
+  /* Make sure the rover isn't pointing into the free block */
+  /* that we just coalesced */
+  if ((rover > (char *)bp) && (rover < NEXT_BLKP(bp)))
+    rover = bp;
   /* $begin mmfree */
   return bp;
 }
@@ -290,16 +296,20 @@
 {
   /* $end mmfirstfit */

-  /* $begin mmfirstfit */
```

```
-   /* First-fit search */
-   void *bp;
-
-   for (bp = heap_listp; GET_SIZE(HDRP(bp)) > 0; bp = NEXT_BLKP(
bp)) {
-     if (!GET_ALLOC(HDRP(bp)) && (asize <= GET_SIZE(HDRP(bp)))) {

-       return bp;
-     }
-   }
-   return NULL; /* No fit */
+   /* Next fit search */
+   char *oldrover = rover;
+
+   /* Search from the rover to the end of list */
+   for ( ; GET_SIZE(HDRP(rover)) > 0; rover = NEXT_BLKP(rover))
+     if (!GET_ALLOC(HDRP(rover)) && (asize <= GET_SIZE(HDRP(rove
r))))
+       return rover;
+
+   /* search from start of list to old rover */
+   for (rover = heap_listp; rover < oldrover; rover = NEXT_BLKP(
rover))
+     if (!GET_ALLOC(HDRP(rover)) && (asize <= GET_SIZE(HDRP(rove
r))))
+       return rover;
+
+   return NULL;  /* no fit found */
 }
 /* $end mmfirstfit */
```

9.18

complete code in `chapter9/code/vm/mm.9.18.c`

```
--- mm.c    2017-11-09 02:57:43.679935907 +0000
+++ mm.9.18.c    2018-02-01 10:08:47.563747327 +0000
@@ -20,7 +20,7 @@
 #define MAX(x, y) ((x) > (y)? (x) : (y))

 /* Pack a size and allocated bit into a word */
-#define PACK(size, alloc)  ((size) | (alloc)) //line:vm:mm:pack
+#define PACK(size, alloc, prev_alloc)  ((size) | (alloc) | (pre
v_alloc << 1)) //line:vm:mm:pack

 /* Read and write a word at address p */
 #define GET(p)       (*(unsigned int *)(p))            //line:v
m:mm:get
@@ -29,6 +29,7 @@
 /* Read the size and allocated fields from address p */
 #define GET_SIZE(p)  (GET(p) & ~0x7)                   //line:v
m:mm:getsize
 #define GET_ALLOC(p) (GET(p) & 0x1)                    //line:v
m:mm:getalloc
+#define GET_PREV_ALLOC(p) ((GET(p) >> 1) & 0x1)

 /* Given block ptr bp, compute address of its header and footer
 */
 #define HDRP(bp)       ((char *)(bp) - WSIZE)
    //line:vm:mm:hdrp
@@ -63,9 +64,9 @@
   if ((heap_listp = mem_sbrk(4*WSIZE)) == (void *)-1) //line:vm
:mm:begininit
     return -1;
   PUT(heap_listp, 0);                          /* Alignment pad
ding */
-  PUT(heap_listp + (1*WSIZE), PACK(DSIZE, 1)); /* Prologue head
er */
-  PUT(heap_listp + (2*WSIZE), PACK(DSIZE, 1)); /* Prologue foot
er */
-  PUT(heap_listp + (3*WSIZE), PACK(0, 1));     /* Epilogue head
```

```
er */
+  PUT(heap_listp + (1*WSIZE), PACK(DSIZE, 1, 1)); /* Prologue h
eader */
+  PUT(heap_listp + (2*WSIZE), PACK(DSIZE, 1, 1)); /* Prologue f
ooter */
+  PUT(heap_listp + (3*WSIZE), PACK(0, 1, 1));     /* Epilogue h
eader */
   heap_listp += (2*WSIZE);                     //line:vm:mm:end
init
   /* $end mminit */

@@ -98,10 +99,10 @@
     return NULL;

   /* Adjust block size to include overhead and alignment reqs.
*/
-  if (size <= DSIZE)                                          /
/line:vm:mm:sizeadjust1
-    asize = 2*DSIZE;                                        //l
ine:vm:mm:sizeadjust2
+  if (size <= WSIZE)
+    asize = DSIZE;
   else
-    asize = DSIZE * ((size + (DSIZE) + (DSIZE-1)) / DSIZE); //l
ine:vm:mm:sizeadjust3
+    asize = DSIZE * ((size + (WSIZE) + (DSIZE-1)) / DSIZE); //l
ine:vm:mm:sizeadjust3

   /* Search the free list for a fit */
   if ((bp = find_fit(asize)) != NULL) {  //line:vm:mm:findfitca
ll
@@ -136,8 +137,16 @@
   }
   /* $begin mmfree */

-  PUT(HDRP(bp), PACK(size, 0));
-  PUT(FTRP(bp), PACK(size, 0));
+  PUT(HDRP(bp), PACK(size, 0, GET_PREV_ALLOC(HDRP(bp))));
+  PUT(FTRP(bp), PACK(size, 0, GET_PREV_ALLOC(HDRP(bp))));
+
```

```
+  if (GET_ALLOC(HDRP(NEXT_BLKP(bp))))
+    PUT(HDRP(NEXT_BLKP(bp)), PACK(GET_SIZE(HDRP(NEXT_BLKP(bp)))
, 1, 0));
+  else {
+    PUT(HDRP(NEXT_BLKP(bp)), PACK(GET_SIZE(HDRP(NEXT_BLKP(bp)))
, 0, 0));
+    PUT(FTRP(NEXT_BLKP(bp)), PACK(GET_SIZE(HDRP(NEXT_BLKP(bp)))
, 0, 0));
+  }
+
   coalesce(bp);
 }

@@ -148,7 +157,7 @@
 /* $begin mmfree */
 static void *coalesce(void *bp)
 {
-  size_t prev_alloc = GET_ALLOC(FTRP(PREV_BLKP(bp)));
+  size_t prev_alloc = GET_PREV_ALLOC(HDRP(bp));
   size_t next_alloc = GET_ALLOC(HDRP(NEXT_BLKP(bp)));
   size_t size = GET_SIZE(HDRP(bp));

@@ -158,22 +167,22 @@

   else if (prev_alloc && !next_alloc) {      /* Case 2 */
     size += GET_SIZE(HDRP(NEXT_BLKP(bp)));
-    PUT(HDRP(bp), PACK(size, 0));
-    PUT(FTRP(bp), PACK(size,0));
+    PUT(HDRP(bp), PACK(size, 0, 1));
+    PUT(FTRP(bp), PACK(size, 0, 1));
   }

   else if (!prev_alloc && next_alloc) {      /* Case 3 */
     size += GET_SIZE(HDRP(PREV_BLKP(bp)));
-    PUT(FTRP(bp), PACK(size, 0));
-    PUT(HDRP(PREV_BLKP(bp)), PACK(size, 0));
+    PUT(FTRP(bp), PACK(size, 0, 1));
+    PUT(HDRP(PREV_BLKP(bp)), PACK(size, 0, 1));
     bp = PREV_BLKP(bp);
   }
```

```
   else {                                     /* Case 4 */
     size += GET_SIZE(HDRP(PREV_BLKP(bp))) +
       GET_SIZE(FTRP(NEXT_BLKP(bp)));
-    PUT(HDRP(PREV_BLKP(bp)), PACK(size, 0));
-    PUT(FTRP(NEXT_BLKP(bp)), PACK(size, 0));
+    PUT(HDRP(PREV_BLKP(bp)), PACK(size, 0, 1));
+    PUT(FTRP(NEXT_BLKP(bp)), PACK(size, 0, 1));
     bp = PREV_BLKP(bp);
   }
   /* $end mmfree */
@@ -246,9 +255,9 @@
     return NULL;                                          //line:
vm:mm:endextend

   /* Initialize free block header/footer and the epilogue heade
r */
-  PUT(HDRP(bp), PACK(size, 0));         /* Free block header */
   //line:vm:mm:freeblockhdr
-  PUT(FTRP(bp), PACK(size, 0));         /* Free block footer */
   //line:vm:mm:freeblockftr
-  PUT(HDRP(NEXT_BLKP(bp)), PACK(0, 1)); /* New epilogue header
*/ //line:vm:mm:newepihdr
+  PUT(HDRP(bp), PACK(size, 0, GET_PREV_ALLOC(HDRP(bp))));
   /* Free block header */   //line:vm:mm:freeblockhdr
+  PUT(FTRP(bp), PACK(size, 0, GET_PREV_ALLOC(HDRP(bp))));
   /* Free block footer */   //line:vm:mm:freeblockftr
+  PUT(HDRP(NEXT_BLKP(bp)), PACK(0, 1, 0)); /* New epilogue head
er */ //line:vm:mm:newepihdr

   /* Coalesce if the previous block was free */
   return coalesce(bp);
 //line:vm:mm:returnblock
@@ -267,15 +276,14 @@
   size_t csize = GET_SIZE(HDRP(bp));

   if ((csize - asize) >= (2*DSIZE)) {
-    PUT(HDRP(bp), PACK(asize, 1));
-    PUT(FTRP(bp), PACK(asize, 1));
+    PUT(HDRP(bp), PACK(asize, 1, 1));
```

```
      bp = NEXT_BLKP(bp);
-     PUT(HDRP(bp), PACK(csize-asize, 0));
-     PUT(FTRP(bp), PACK(csize-asize, 0));
+     PUT(HDRP(bp), PACK(csize-asize, 0, 1));
+     PUT(FTRP(bp), PACK(csize-asize, 0, 1));
    }
    else {
-     PUT(HDRP(bp), PACK(csize, 1));
-     PUT(FTRP(bp), PACK(csize, 1));
+     PUT(HDRP(bp), PACK(csize, 1, 1));
+     PUT(HDRP(NEXT_BLKP(bp)), PACK(csize, 1, 1));
    }
  }
  /* $end mmplace */
```

9.19

1)

a: Right. 2^(k+1) block size for (2^k)+1 allocation request

b: wrong.

c: wrong. LIFO is also fast.

d: wrong. almost every strategy has external fragmentation problem.

2)

a: wrong. first fit should be fast in this condition

b: wrong. should be order by block size

c: wrong. min spare size not max

d: Right.

3)

b

ref 9.10.3

9.20

using malloc lib code from csapp site

- implicit idle list
- allocated block with header & footer
- idle block with header & footer
- no GC
- first fit strategy

another modification to malloc lib file

mm.h

```
--- ../vm/mm.h    2017-11-09 02:57:43.679935907 +0000
+++ mm.h    2017-11-09 02:57:43.679935907 +0000
@@ -5,6 +5,9 @@
 extern void mm_free (void *ptr);
 /* $end mallocinterface */

+#define malloc(size) mm_malloc(size)
+#define free(ptr) mm_free(ptr)
+
 extern void *mm_realloc(void *ptr, size_t size);
 extern void *mm_calloc (size_t nmemb, size_t size);
 extern void mm_checkheap(int verbose);
```

memlib.c

```
--- ../vm/memlib.c    2017-11-09 02:57:43.679935907 +0000
+++ memlib.c    2017-11-09 02:57:43.675935936 +0000
@@ -15,23 +15,18 @@
 #include "csapp.h"
 #include "memlib.h"

-#define MAX_HEAP (20*(1<<20))  /* 20 MB */
-
-
 /* $begin memlib */
 /* Private global variables */
 static char *mem_heap;     /* Points to first byte of heap */
 static char *mem_brk;      /* Points to last byte of heap plus
1 */
-static char *mem_max_addr; /* Max legal heap addr plus 1*/

 /*
  * mem_init - Initialize the memory system model
  */
 void mem_init(void)
 {
-  mem_heap = (char *)Malloc(MAX_HEAP);
+  mem_heap = (char *)sbrk(0);
   mem_brk = (char *)mem_heap;
-  mem_max_addr = (char *)(mem_heap + MAX_HEAP);
 }

 /*
@@ -43,7 +38,7 @@
 {
   char *old_brk = mem_brk;

-  if ( (incr < 0) || ((mem_brk + incr) > mem_max_addr)) {
+  if ( (incr < 0) || ((mem_brk = sbrk(incr)) == (void *)-1)) {
     errno = ENOMEM;
     fprintf(stderr, "ERROR: mem_sbrk failed. Ran out of memory.
..\n");
     return (void *)-1;
```

main.c file measure malloc performance

```
/*
 * main.c
 */
#include <stdio.h>
#include "csapp.h"

#ifdef CUS_MALLOC
#include "mm.h"
#include "memlib.h"
#else
#include <stdlib.h>
#endif

#define LOOP 10000

int main(int argc, char* argv[]) {
  void* m_start = sbrk(0);
  size_t malloc_size = 0;

  int i;
  for (i = 0; i < LOOP; i+=2) {
    void* ptr_f = malloc(i);
    void* ptr = malloc(i+1);
    free(ptr_f);

    malloc_size += i+1;
  }

  void* m_end = sbrk(0);
  size_t heap_size = (size_t)(m_end - m_start);

  printf("malloc size: %ld, heap_size: %ld\n", malloc_size, heap
_size);

  return 0;
}
```

run `make` to generate both origin main executable file and custom version(using `-DCUS_MALLOC`)

```
CC = gcc
CFLAGS = -m64 -pthread -DCUS_MALLOC
SRCS = mm.c memlib.c csapp.c

all: origin custom diff

measure:
	time ./origin.main
	time ./custom.main

origin:
	$(CC) -m64 main.c -o origin.main

custom:
	$(CC) $(CFLAGS) $(SRCS) main.c -o custom.main

diff:
	(diff -u ../vm/mm.h mm.h > mm.h.diff; cd .)
	(diff -u ../vm/memlib.c memlib.c > memlib.c.diff; cd .)

test:

.PHONY: clean
clean:
	find . -type f -executable -print0 | xargs -0 rm -f --
```

measurement

```
(cd chapter9/code/malloc; make measure)

time ./origin.main
malloc size: 25000000, heap_size: 28311552
0.00user 0.01system 0:00.01elapsed 100%CPU (0avgtext+0avgdata 19
256maxresident)k
0inputs+0outputs (0major+4547minor)pagefaults 0swaps

time ./custom.main
malloc size: 25000000, heap_size: 31327104
0.58user 0.00system 0:00.58elapsed 99%CPU (0avgtext+0avgdata 305
92maxresident)k
0inputs+0outputs (0major+7339minor)pagefaults 0swaps
```

System-Level I/O

> I think the major good idea in Unix was its clean and simple interface: open, close, read, and write.
>
> by Ken Thompson

10.1 - 10.5 visit book

10.6 - 10.10 visit here

test

code directory: `./code`

test way:

- assert means assert function from `<assert.h>`
- output means to watch code output to judge if it works right

solution	code file	test way
10.6	10.6.c	output
10.7	10.7.c	output
10.8	10.8.c	output
10.9	----	----
10.10	10.10.c	output

10.6

```
fd = 4
```

10.7

```c
/*
 * 10.7.c
 */
#include <stdio.h>
#include "csapp.h"

int main(int argc, char* argv[]) {
  int n;
  char buf[MAXBUF];

  while ((n = Rio_readn(STDIN_FILENO, buf, MAXBUF)) != 0)
    Rio_writen(STDOUT_FILENO, buf, n);

  return 0;
}
```

10.8

```c
/*
 * 10.8.c
 */
#include <stdio.h>
#include "csapp.h"


int main(int argc, char* argv[]) {
  struct stat stat;
  char *type, *readok;

  int fd;
  if (argc <= 1)
    fd = 0; // stdin
  else
    fd = atoi(argv[1]);

  Fstat(fd, &stat);

  if (S_ISREG(stat.st_mode))
    type = "regular";
  else if (S_ISDIR(stat.st_mode))
    type = "dir";
  else
    type = "other";
  if ((stat.st_mode & S_IRUSR))
    readok = "yes";
  else
    readok = "no";

  printf("type: %s, read: %s\n", type, readok);

  return 0;
}
```

10.9

```
if (Fork() == 0) {
  Dup2(0, 3);
  Execve("fstatcheck", argv, envp);
}
```

10.10

```c
/*
 * 10.10.c
 */
#include <stdio.h>
#include "csapp.h"

int main(int argc, char* argv[]) {
  int n;
  rio_t rio;
  char buf[MAXLINE];

  if (argc == 2) {
    int fd = Open(argv[1], O_RDONLY, 0);
    while ((n = Rio_readn(fd, buf, MAXBUF)) != 0)
      Rio_writen(STDOUT_FILENO, buf, n);
    exit(0);
  }

  Rio_readinitb(&rio, STDIN_FILENO);
  while ((n = Rio_readlineb(&rio, buf, MAXLINE)) != 0)
    Rio_writen(STDOUT_FILENO, buf, n);

  return 0;
}
```

Network Programming

> 640k is enough for anyone, and by the way, what's a network?
>
> by William Gates III

11.1 - 11.5 visit book

11.6 - 11.13 visit here

test

code directory: `./code`

test way:

- browser means start server and use browser visit server and watch result

solution	code file	test way
11.6	tiny.6.c	browser
11.7	tiny.7.c	browser
11.8	tiny.8.c	browser
11.9	tiny.9.c	browser
11.10	cgi-bin/form-adder.c, tiny.origin.c	browser
11.11	cgi-bin/head-adder.c, tiny.11.c	browser
11.12	cgi-bin/post-adder.c, tiny.12.c	browser
11.13	tiny.13.c	browser

11.6

A.

```
--- tiny.origin.c    2017-11-09 02:57:43.651936112 +0000
+++ tiny.6.c    2017-11-09 02:57:43.651936112 +0000
@@ -13,6 +13,8 @@
 void clienterror(int fd, char *cause, char *errnum,
     char *shortmsg, char *longmsg);

+void echo(int connfd);
+
 int main(int argc, char **argv)
 {
   int listenfd, connfd;
@@ -33,11 +35,24 @@
     Getnameinfo((SA *) &clientaddr, clientlen, hostname, MAXLIN
E,
         port, MAXLINE, 0);
     printf("Accepted connection from (%s, %s)\n", hostname, por
t);
-    doit(connfd);                                               /
/line:netp:tiny:doit
+    echo(connfd);
     Close(connfd);                                              /
/line:netp:tiny:close
   }
 }

+void echo(int connfd) {
+  size_t n;
+  char buf[MAXLINE];
+  rio_t rio;
+
+  Rio_readinitb(&rio, connfd);
+  while ((n = Rio_readlineb(&rio, buf, MAXLINE)) != 0) {
+    if (strcmp(buf, "\r\n") == 0)
+      break;
+    Rio_writen(connfd, buf, n);
+  }
```

```
+}
+
 /*
  * doit - handle one HTTP request/response transaction
  */
```

B.

`./tiny.6 5000` run server and firefox visit localhost:5000

```
GET / HTTP/1.1
Host: localhost:5000
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:53.0) Gec
ko/20100101 Firefox/53.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/
*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

C.

HTTP 1.1

D.

visit rfc section 14 Header Field Definitions

Accept: 14.1

The Accept request-header field can be used to specify certain media types which are acceptable for the response. Accept headers can be used to indicate that the request is specifically limited to a small set of desired types, as in the case of a request for an in-line image.

Accept-Encoding: 14.3

The Accept-Encoding request-header field is similar to Accept, but restricts the content-codings that are acceptable in the response.

Accept-Language: 14.4

The Accept-Language request-header field is similar to Accept, but restricts the set of natural languages that are preferred as a response to the request. Language tags are defined in section 3.10.

Connection: 14.10

The Connection general-header field allows the sender to specify options that are desired for that particular connection and MUST NOT be communicated by proxies over further connections.

Host: 14.23

The Host request-header field specifies the Internet host and port number of the resource being requested, as obtained from the original URI given by the user or referring resource. The Host field value MUST represent the naming authority of the origin server or gateway given by the original URL. This allows the origin server or gateway to differentiate between internally-ambiguous URLs, such as the root “/” URL of a server for multiple host names on a single IP address.

User-Agent: 14.43

The User-Agent request-header field contains information about the user agent originating the request. This is for statistical purposes, the tracing of protocol violations, and automated recognition of user agents for the sake of tailoring responses to avoid particular user agent limitations. User agents SHOULD include this field with requests. The field can contain multiple product tokens (section 3.8) and comments identifying the agent and any subproducts which form a significant part of the user agent. By convention, the product tokens are listed in order of their significance for identifying the application.

11.7

visit mdn to check all mime type

```
--- tiny.origin.c    2017-11-09 02:57:43.651936112 +0000
+++ tiny.7.c    2017-11-09 02:57:43.651936112 +0000
@@ -173,6 +173,8 @@
     strcpy(filetype, "image/png");
   else if (strstr(filename, ".jpg"))
     strcpy(filetype, "image/jpeg");
+  else if (strstr(filename, ".mpeg"))
+    strcpy(filetype, "video/mpeg");
   else
     strcpy(filetype, "text/plain");
 }
```

run server

```
(cd chapter11/code; make && ./tiny.7 5000)
```

browser visit

```
http://localhost:5000/ghost-in-shell.mpeg
```

11.8

```
--- tiny.origin.c    2017-11-09 02:57:43.651936112 +0000
+++ tiny.8.c    2017-11-09 02:57:43.651936112 +0000
@@ -12,6 +12,7 @@
 void serve_dynamic(int fd, char *filename, char *cgiargs);
 void clienterror(int fd, char *cause, char *errnum,
     char *shortmsg, char *longmsg);
+void sigchild_handler(int sig);

 int main(int argc, char **argv)
 {
@@ -26,6 +27,9 @@
     exit(1);
   }

+  if (Signal(SIGCHLD, sigchild_handler) == SIG_ERR)
+    unix_error("signal child handler error");
+
   listenfd = Open_listenfd(argv[1]);
   while (1) {
     clientlen = sizeof(clientaddr);
@@ -38,6 +42,15 @@
   }
 }

+void sigchild_handler(int sig) {
+  int old_errno = errno;
+  int status;
+  pid_t pid;
+  while ((pid = waitpid(-1, &status, WNOHANG)) > 0) {
+  }
+  errno = old_errno;
+}
+
 /*
  * doit - handle one HTTP request/response transaction
  */
@@ -196,7 +209,6 @@
     Dup2(fd, STDOUT_FILENO);         /* Redirect stdout to clie
```

```
nt */ //line:netp:servedynamic:dup2
     Execve(filename, emptylist, environ); /* Run CGI program */
 //line:netp:servedynamic:execve
   }
-  Wait(NULL); /* Parent waits for and reaps child */ //line:net
p:servedynamic:wait
 }

 /*
```

11.9

```
--- tiny.origin.c    2017-11-09 02:57:43.651936112 +0000
+++ tiny.9.c    2017-11-09 02:57:43.651936112 +0000
@@ -152,12 +152,12 @@
   printf("Response headers:\n");
   printf("%s", buf);

-  /* Send response body to client */
   srcfd = Open(filename, O_RDONLY, 0);    //line:netp:servestat
ic:open
-  srcp = Mmap(0, filesize, PROT_READ, MAP_PRIVATE, srcfd, 0);//
line:netp:servestatic:mmap
+  srcp = (char*)Malloc(filesize);
+  Rio_readn(srcfd, srcp, filesize);
   Close(srcfd);                           //line:netp:servestat
ic:close
   Rio_writen(fd, srcp, filesize);         //line:netp:servestat
ic:write
-  Munmap(srcp, filesize);                 //line:netp:servestat
ic:munmap
+  free(srcp);
 }

 /*
```

11.10

A.

```html
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<title>Tiny Server</title>
</head>
<body>
<form action="/cgi-bin/form-adder" method="GET">
  <p>first number: <input type="text" name="first"/></p>
  <p>second number: <input type="text" name="second"/></p>
  <input type="submit" value="Submit"/>
</form>
</body>
</html>
```

browser should visit `localhost:5000/cgi-bin/form-adder?first=222&second=333`

B.

`form-adder` handle query string like `first=ddd&second=dddd`

```
--- adder.c    2017-11-09 02:57:43.647936141 +0000
+++ form-adder.c    2017-11-09 02:57:43.647936141 +0000
@@ -1,5 +1,5 @@
 /*
- * adder.c - a minimal CGI program that adds two numbers togeth
er
+ * form-adder.c - a minimal CGI program that adds two numbers t
ogether
  */
 #include "../csapp.h"

@@ -12,10 +12,8 @@
   if ((buf = getenv("QUERY_STRING")) != NULL) {
     p = strchr(buf, '&');
     *p = '\0';
-    strcpy(arg1, buf);
-    strcpy(arg2, p+1);
-    n1 = atoi(arg1);
-    n2 = atoi(arg2);
+    sscanf(buf, "first=%d", &n1);
+    sscanf(p+1, "second=%d", &n2);
   }

   /* Make the response body */
```

11.11

from rfc2626 section 9.4 HEAD

The HEAD method is identical to GET except that the server MUST NOT return a message-body in the response.

tiny.c changes

```
--- tiny.origin.c    2017-11-09 02:57:43.651936112 +0000
+++ tiny.11.c    2017-11-09 02:57:43.651936112 +0000
@@ -7,9 +7,9 @@
 void doit(int fd);
 void read_requesthdrs(rio_t *rp);
 int parse_uri(char *uri, char *filename, char *cgiargs);
-void serve_static(int fd, char *filename, int filesize);
+void serve_static(int fd, char *filename, int filesize, char *m
ethod);
 void get_filetype(char *filename, char *filetype);
-void serve_dynamic(int fd, char *filename, char *cgiargs);
+void serve_dynamic(int fd, char *filename, char *cgiargs, char
*method);
 void clienterror(int fd, char *cause, char *errnum,
     char *shortmsg, char *longmsg);

@@ -55,7 +55,7 @@
     return;
   printf("%s", buf);
   sscanf(buf, "%s %s %s", method, uri, version);       //line:n
etp:doit:parserequest
-  if (strcasecmp(method, "GET")) {                     //line:n
etp:doit:beginrequesterr
+  if (!(strcasecmp(method, "GET") == 0 || strcasecmp(method, "H
EAD") == 0)) {
     clienterror(fd, method, "501", "Not Implemented",
         "Tiny does not implement this method");
     return;
@@ -76,7 +76,7 @@
           "Tiny couldn't read the file");
       return;
     }
```

```
-    serve_static(fd, filename, sbuf.st_size);        //line:net
p:doit:servestatic
+    serve_static(fd, filename, sbuf.st_size, method);        //
line:netp:doit:servestatic
   }
   else { /* Serve dynamic content */
     if (!(S_ISREG(sbuf.st_mode)) || !(S_IXUSR & sbuf.st_mode))
{ //line:netp:doit:executable
@@ -84,7 +84,7 @@
           "Tiny couldn't run the CGI program");
       return;
     }
-    serve_dynamic(fd, filename, cgiargs);            //line:net
p:doit:servedynamic
+    serve_dynamic(fd, filename, cgiargs, method);            //
line:netp:doit:servedynamic
   }
 }

@@ -136,7 +136,7 @@
 /*
  * serve_static - copy a file back to the client
  */
-void serve_static(int fd, char *filename, int filesize)
+void serve_static(int fd, char *filename, int filesize, char *m
ethod)
 {
   int srcfd;
   char *srcp, filetype[MAXLINE], buf[MAXBUF];
@@ -152,6 +152,9 @@
   printf("Response headers:\n");
   printf("%s", buf);

+  if (strcasecmp(method, "HEAD") == 0)
+    return;
+
   /* Send response body to client */
   srcfd = Open(filename, O_RDONLY, 0);    //line:netp:servestat
ic:open
   srcp = Mmap(0, filesize, PROT_READ, MAP_PRIVATE, srcfd, 0);//
```

```
line:netp:servestatic:mmap
@@ -180,7 +183,7 @@
 /*
  * serve_dynamic - run a CGI program on behalf of the client
  */
-void serve_dynamic(int fd, char *filename, char *cgiargs)
+void serve_dynamic(int fd, char *filename, char *cgiargs, char
*method)
 {
   char buf[MAXLINE], *emptylist[] = { NULL };

@@ -193,6 +196,7 @@
   if (Fork() == 0) { /* Child */ //line:netp:servedynamic:fork
     /* Real server would set all CGI vars here */
     setenv("QUERY_STRING", cgiargs, 1); //line:netp:servedynami
c:setenv
+    setenv("REQUEST_METHOD", method, 1);
     Dup2(fd, STDOUT_FILENO);         /* Redirect stdout to clie
nt */ //line:netp:servedynamic:dup2
     Execve(filename, emptylist, environ); /* Run CGI program */
 //line:netp:servedynamic:execve
   }
```

adder.c changes

```
--- adder.c    2017-11-09 02:57:43.647936141 +0000
+++ head-adder.c    2017-11-09 02:57:43.647936141 +0000
@@ -1,10 +1,10 @@
 /*
- * adder.c - a minimal CGI program that adds two numbers togeth
er
+ * head-adder.c - a minimal CGI program that adds two numbers t
ogether
  */
 #include "../csapp.h"

 int main(void) {
-  char *buf, *p;
+  char *buf, *p, *method;
```

```
    char arg1[MAXLINE], arg2[MAXLINE], content[MAXLINE];
    int n1=0, n2=0;

@@ -18,6 +18,8 @@
      n2 = atoi(arg2);
    }

+   method = getenv("REQUEST_METHOD");
+
    /* Make the response body */
    sprintf(content, "Welcome to add.com: ");
    sprintf(content, "%sTHE Internet addition portal.\r\n<p>", co
ntent);
@@ -29,7 +31,10 @@
    printf("Connection: close\r\n");
    printf("Content-length: %d\r\n", (int)strlen(content));
    printf("Content-type: text/html\r\n\r\n");
-   printf("%s", content);
+
+   if (strcasecmp(method, "HEAD") != 0)
+     printf("%s", content);
+
    fflush(stdout);

    exit(0);
```

11.12

run server

```
(cd chapter11/code; make && ./tiny.12 5000)
```

visit `http://localhost:5000/post-home.html` and submit

POST method pass param by message-body behind Headers part. When we input 9 and 10 and submit, socket pass content

```
POST /cgi-bin/post-adder HTTP/1.1
Host: localhost:5000
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:53.0) Gec
ko/20100101 Firefox/53.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/
*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 16
Referer: http://localhost:5000/post-home.html
Cookie: _ga=GA1.1.1286836072.1494744693
Connection: keep-alive
Upgrade-Insecure-Requests: 1
(CRLF here, message-body below)
first=9&second=10
```

Content-Length marks the length of message-body. use `readnb` to fetch post param.

```
--- tiny.origin.c    2017-11-09 02:57:43.651936112 +0000
+++ tiny.12.c    2017-11-09 02:57:43.651936112 +0000
@@ -5,7 +5,7 @@
 #include "csapp.h"

 void doit(int fd);
-void read_requesthdrs(rio_t *rp);
+int read_requesthdrs(rio_t *rp, char *method);
```

```
 int parse_uri(char *uri, char *filename, char *cgiargs);
 void serve_static(int fd, char *filename, int filesize);
 void get_filetype(char *filename, char *filetype);
@@ -55,12 +55,14 @@
     return;
   printf("%s", buf);
   sscanf(buf, "%s %s %s", method, uri, version);       //line:n
etp:doit:parserequest
-  if (strcasecmp(method, "GET")) {                     //line:n
etp:doit:beginrequesterr
+  if (!(strcasecmp(method, "GET") == 0 || strcasecmp(method, "P
OST") == 0)) {
     clienterror(fd, method, "501", "Not Implemented",
         "Tiny does not implement this method");
     return;
   }                                                    //line:n
etp:doit:endrequesterr
-  read_requesthdrs(&rio);                              //line:n
etp:doit:readrequesthdrs
+  int param_len = read_requesthdrs(&rio, method);
+
+  Rio_readnb(&rio, buf, param_len);

   /* Parse URI from GET request */
   is_static = parse_uri(uri, filename, cgiargs);       //line:n
etp:doit:staticcheck
@@ -84,24 +86,29 @@
           "Tiny couldn't run the CGI program");
       return;
     }
-    serve_dynamic(fd, filename, cgiargs);            //line:net
p:doit:servedynamic
+    if (strcasecmp(method, "GET") == 0)
+      serve_dynamic(fd, filename, cgiargs);
+    else
+      serve_dynamic(fd, filename, buf);
   }
 }

 /*
```

```
  * read_requesthdrs - read HTTP request headers
  */
-void read_requesthdrs(rio_t *rp)
+int read_requesthdrs(rio_t *rp, char *method)
 {
   char buf[MAXLINE];
+  int len = 0;

-  Rio_readlineb(rp, buf, MAXLINE);
-  printf("%s", buf);
-  while(strcmp(buf, "\r\n")) {          //line:netp:readhdrs:ch
eckterm
+  do {
     Rio_readlineb(rp, buf, MAXLINE);
     printf("%s", buf);
-  }
-  return;
+    if (strcasecmp(method, "POST") == 0 && strncasecmp(buf, "Co
ntent-Length:", 15) == 0)
+      sscanf(buf, "Content-Length: %d", &len);
+  } while(strcmp(buf, "\r\n"));
+
+  return len;
 }
 /*
  * parse_uri - parse URI into filename and CGI args
```

post-adder code

```
/*
 * post-adder.c - a minimal CGI program that adds two numbers to
gether
 */
#include "../csapp.h"

int main(void) {
  char *buf, *p;
  char arg1[MAXLINE], arg2[MAXLINE], content[MAXLINE];
  int n1=0, n2=0;

  /* Extract the two arguments */
  if ((buf = getenv("QUERY_STRING")) != NULL) {
    p = strchr(buf, '&');
    *p = '\0';
    sscanf(buf, "first=%d", &n1);
    sscanf(p+1, "second=%d", &n2);
  }

  /* Make the response body */
  sprintf(content, "Welcome to add.com: ");
  sprintf(content, "%sTHE Internet addition portal.\r\n<p>", con
tent);
  sprintf(content, "%sThe answer is: %d + %d = %d\r\n<p>",
      content, n1, n2, n1 + n2);
  sprintf(content, "%sThanks for visiting!\r\n", content);

  /* Generate the HTTP response */
  printf("Connection: close\r\n");
  printf("Content-length: %d\r\n", (int)strlen(content));
  printf("Content-type: text/html\r\n\r\n");
  printf("%s", content);
  fflush(stdout);

  exit(0);
}
```

11.13

```
--- tiny.origin.c    2017-11-09 02:57:43.651936112 +0000
+++ tiny.13.c    2017-11-09 02:57:43.651936112 +0000
@@ -13,6 +13,17 @@
 void clienterror(int fd, char *cause, char *errnum,
     char *shortmsg, char *longmsg);

+// improved rio written
+void Im_rio_writen(int fd, void *usrbuf, size_t n) {
+  if (rio_writen(fd, usrbuf, n) != n) {
+    if (errno == EPIPE)
+      fprintf(stderr, "EPIPE error");
+
+    fprintf(stderr, "%s ", strerror(errno));
+    unix_error("client side has ended connection");
+  }
+}
+
 int main(int argc, char **argv)
 {
   int listenfd, connfd;
@@ -26,6 +37,9 @@
     exit(1);
   }

+  if (Signal(SIGPIPE, SIG_IGN) == SIG_ERR)
+    unix_error("mask signal pipe error");
+
   listenfd = Open_listenfd(argv[1]);
   while (1) {
     clientlen = sizeof(clientaddr);
@@ -148,7 +162,7 @@
   sprintf(buf, "%sConnection: close\r\n", buf);
   sprintf(buf, "%sContent-length: %d\r\n", buf, filesize);
   sprintf(buf, "%sContent-type: %s\r\n\r\n", buf, filetype);
-  Rio_writen(fd, buf, strlen(buf));       //line:netp:servestatic:endserve
+  Im_rio_writen(fd, buf, strlen(buf));       //line:netp:servestatic:endserve
```

```
   printf("Response headers:\n");
   printf("%s", buf);

@@ -156,7 +170,7 @@
   srcfd = Open(filename, O_RDONLY, 0);    //line:netp:servestat
ic:open
   srcp = Mmap(0, filesize, PROT_READ, MAP_PRIVATE, srcfd, 0);//
line:netp:servestatic:mmap
   Close(srcfd);                           //line:netp:servestat
ic:close
-  Rio_writen(fd, srcp, filesize);         //line:netp:servestat
ic:write
+  Im_rio_writen(fd, srcp, filesize);         //line:netp:serves
tatic:write
   Munmap(srcp, filesize);                 //line:netp:servestat
ic:munmap
 }

@@ -186,11 +200,13 @@

   /* Return first part of HTTP response */
   sprintf(buf, "HTTP/1.0 200 OK\r\n");
-  Rio_writen(fd, buf, strlen(buf));
+  Im_rio_writen(fd, buf, strlen(buf));
   sprintf(buf, "Server: Tiny Web Server\r\n");
-  Rio_writen(fd, buf, strlen(buf));
+  Im_rio_writen(fd, buf, strlen(buf));

   if (Fork() == 0) { /* Child */ //line:netp:servedynamic:fork
+    if (Signal(SIGPIPE, SIG_DFL) == SIG_ERR)
+      unix_error("unmask signal pipe error");
     /* Real server would set all CGI vars here */
     setenv("QUERY_STRING", cgiargs, 1); //line:netp:servedynami
c:setenv
     Dup2(fd, STDOUT_FILENO);         /* Redirect stdout to clie
nt */ //line:netp:servedynamic:dup2
@@ -216,10 +232,10 @@

   /* Print the HTTP response */
   sprintf(buf, "HTTP/1.0 %s %s\r\n", errnum, shortmsg);
```

```
-   Rio_writen(fd, buf, strlen(buf));
+   Im_rio_writen(fd, buf, strlen(buf));
    sprintf(buf, "Content-type: text/html\r\n");
-   Rio_writen(fd, buf, strlen(buf));
+   Im_rio_writen(fd, buf, strlen(buf));
    sprintf(buf, "Content-length: %d\r\n\r\n", (int)strlen(body))
;
-   Rio_writen(fd, buf, strlen(buf));
-   Rio_writen(fd, body, strlen(body));
+   Im_rio_writen(fd, buf, strlen(buf));
+   Im_rio_writen(fd, body, strlen(body));
 }
```

ignore SIGPIPE and show more friendly errer message.

unmask SIGPIPE, leave child process to handle it.

Concurrent Programming

> Multi-tasking arises out of distraction itself.
>
> by Marilyn vos Savant

12.1 - 12.15 visit book

12.16 - 12.39 visit here

test

prerequisite

- need wrk to benchmark server in 35, 36, 37, 38
- need package `apache2-utils` (required by command line `ab`) to benchmark proxy server

code directory: `./code`

test way:

- output means to watch code output to judge if it works right
- benchmark means using wrk/ab to make lots of connections at same time to test server

solution	code file	test way
12.16	12.16.c	output
12.17	12.17.c	output
12.18	----	----
12.19	12.19.c	output
12.20	12.20.c	output
12.21	12.21.c	output
12.22	12.22.c	using telnet, more visit 12.22.md
12.23	12.23.c	visit 12.23.md
12.24	----	----
12.25	----	----
12.26	12.26.c	output
12.27	----	----
12.28	----	----
12.29	----	----
12.30	----	----
12.31	12.31.c	run and input or BOOM after 5 seconds
12.32	12.32.c	run and input or BOOM after 5 seconds
12.33	12.33.c	run and input or BOOM after 5 seconds
12.34	12.34.c	visit 12.34.md, how to measure performance
12.35	12.35.c	benchmark
12.36	12.36/*	benchmark
12.37	12.37.c	benchmark
12.38	12.38/*	benchmark
12.39	12.39/*	benchmark, more see 12.39.md

12.16

```c
/*
 * 12.16.c
 */
#include <stdio.h>
#include "csapp.h"

void *thread(void *vargp);

#define DEFAULT 4

int main(int argc, char* argv[]) {
  int N;
  if (argc > 2)
    unix_error("too many param");
  else if (argc == 2)
    N = atoi(argv[1]);
  else
    N = DEFAULT;

  int i;
  pthread_t tid;
  for (i = 0; i < N; i++) {
    Pthread_create(&tid, NULL, thread, NULL);
  }
  Pthread_exit(NULL);
}

void *thread(void *vargp) {
  printf("Hello, world\n");
  return NULL;
}
```

12.17

A.

main thread didn't wait other thread.

B.

pthread_exit

```
/*
 * 12.17.c
 */
#include "csapp.h"
void *thread(void *vargp);

int main()
{
  pthread_t tid;

  Pthread_create(&tid, NULL, thread, NULL);
  // exit(0);
  Pthread_exit(NULL);
}

/* Thread routine */
void *thread(void *vargp)
{
  Sleep(1);
  printf("Hello, world!\n");
  return NULL;
}
```

12.18

A unsafe

B safe

C unsafe

12.19

pay attention to `static int reader_first`

```
/*
 * 12.19.c
 */
#include <stdio.h>
#include "csapp.h"

#define WRITE_LIMIT 100000
#define PEOPLE 4

static int readtimes;
static int writetimes;
static int readcnt;
// if a reader is waiting when writing, reader first next round
static int reader_first;
sem_t mutex, w;

void *reader(void *vargp) {
  while (1) {
    P(&mutex);
    readcnt++;
    if (readcnt == 1)
      P(&w);
    V(&mutex);

    /* Critical section */
    readtimes++;
    reader_first = 0;
    /* Critical section */

    P(&mutex);
    readcnt--;
    if (readcnt == 0)
      V(&w);
    V(&mutex);
  }
}
```

```
void *writer(void *vargp) {
  while (1) {
    if (reader_first == 1)
      continue;

    P(&w);

    /* Critical section */
    writetimes++;
    if (writetimes == WRITE_LIMIT) {
      printf("read/write: %d/%d\n", readtimes, writetimes);
      exit(0);
    }
    /* Critical section */

    // if a reader is waiting, reader first next round
    if (readcnt == 1)
      reader_first = 1;
    V(&w);
  }
}

void init(void) {
  readcnt = 0;
  readtimes = 0;
  writetimes = 0;
  reader_first = 0;
  Sem_init(&w, 0, 1);
  Sem_init(&mutex, 0, 1);
}


int main(int argc, char* argv[]) {
  int i;
  pthread_t tid;

  init();

  for (i = 0; i < PEOPLE; i++)
```

```
    if (i%2 == 0)
      Pthread_create(&tid, NULL, reader, NULL);
    else
      Pthread_create(&tid, NULL, writer, NULL);

  Pthread_exit(NULL);
  exit(0);
}
```

12.20

`readercnt` limits the readers number at the same time.

```
/*
 * 12.20.c
 */
#include <stdio.h>
#include "csapp.h"

#define WRITE_LIMIT 100000
#define PEOPLE 20 // 10 reader and 10 writer
#define N 5

static int readtimes;
static int writetimes;
sem_t mutex;
sem_t readercnt;

void *reader(void *vargp) {
  while (1) {
    P(&readercnt);
    P(&mutex);

    readtimes++;

    V(&mutex);
    V(&readercnt);
  }
}

void *writer(void *vargp) {
  while (1) {
    P(&mutex);

    writetimes++;
    if (writetimes == WRITE_LIMIT) {
      printf("read/write: %d/%d\n", readtimes, writetimes);
      exit(0);
    }
```

```
    V(&mutex);
  }
}

void init(void) {
  readtimes = 0;
  writetimes = 0;
  Sem_init(&mutex, 0, 1);
  Sem_init(&readercnt, 0, N);
}


int main(int argc, char* argv[]) {
  int i;
  pthread_t tid;

  init();

  for (i = 0; i < PEOPLE; i++) {
    if (i%2 == 0)
      Pthread_create(&tid, NULL, reader, NULL);
    else
      Pthread_create(&tid, NULL, writer, NULL);
  }

  Pthread_exit(NULL);
  exit(0);
}
```

12.21

`writecnt` record how many writers are waiting.

```
/*
 * 12.21.c
 */
#include <stdio.h>
#include "csapp.h"

#define WRITE_LIMIT 100000
#define PEOPLE 4

static int readtimes;
static int writetimes;
static int writecnt;
sem_t mutex, w;

static int number;

void *reader(void *vargp) {
  while (1) {
    // writer first
    if (writecnt > 0)
      continue;

    P(&w);

    /* Critical section */
    readtimes++;
    /* Critical section */

    V(&w);
  }
}

void *writer(void *vargp) {
  while (1) {
    P(&mutex);
    // one more writer wait to write
```

```
    writecnt++;
    V(&mutex);

    P(&w);

    /* Critical section */
    writetimes++;
    if (writetimes == WRITE_LIMIT) {
      printf("read/write: %d/%d\n", readtimes, writetimes);
      exit(0);
    }
    /* Critical section */

    V(&w);

    P(&mutex);
    // writer has written
    writecnt--;
    V(&mutex);
  }
}

void init(void) {
  writecnt = 0;
  readtimes = 0;
  writetimes = 0;
  Sem_init(&w, 0, 1);
  Sem_init(&mutex, 0, 1);
}


int main(int argc, char* argv[]) {
  int i;
  pthread_t tid;

  init();

  for (i = 0; i < PEOPLE; i++) {
    if (i%2 == 0)
      Pthread_create(&tid, NULL, reader, NULL);
```

```
    else
      Pthread_create(&tid, NULL, writer, NULL);
  }

  Pthread_exit(NULL);
  exit(0);
}
```

12.22

```
#include "csapp.h"

/* read line from connfd and echo line to connfd */
int echo_line(int connfd);

void command(void);

int main(int argc, char **argv)
{
  int listenfd, connfd;
  socklen_t clientlen;
  struct sockaddr_storage clientaddr;
  fd_set read_set, ready_set;

  if (argc != 2) {
    fprintf(stderr, "usage: %s <port>\nuse port 5000 here\n", argv[0]);
    // default port 5000
    listenfd = Open_listenfd("5000");
  } else {
    listenfd = Open_listenfd(argv[1]);  //line:conc:select:openlistenfd
  }

  FD_ZERO(&read_set);              /* Clear read set */ //line:conc:select:clearreadset
  FD_SET(STDIN_FILENO, &read_set); /* Add stdin to read set */ //line:conc:select:addstdin
  FD_SET(listenfd, &read_set);     /* Add listenfd to read set */ //line:conc:select:addlistenfd

  // max n for select
  int n = listenfd+1;

  while (1) {
    ready_set = read_set;
    Select(n, &ready_set, NULL, NULL, NULL); //line:conc:select:select
```

```
    if (FD_ISSET(STDIN_FILENO, &ready_set)) //line:conc:select:s
tdinready
      command(); /* Read command line from stdin */

    if (FD_ISSET(listenfd, &ready_set)) { //line:conc:select:lis
tenfdready
      clientlen = sizeof(struct sockaddr_storage);
      connfd = Accept(listenfd, (SA *)&clientaddr, &clientlen);


      // listen to accepted io ports
      if (connfd+1 > FD_SETSIZE) {
        fprintf(stderr, "too many clients\n");
        Close(connfd);
      }
      n = n > connfd+1 ? n : connfd+1;
      FD_SET(connfd, &read_set);
    }

    // echo one line every time
    int fd;
    for (fd = listenfd+1; fd < n; fd++)
      if (FD_ISSET(fd, &ready_set))
        if (echo_line(fd) == -1) {
          Close(fd);
          FD_CLR(fd, &read_set);
        }
  }
}

void command(void) {
  char buf[MAXLINE];
  if (!Fgets(buf, MAXLINE, stdin))
    exit(0); /* EOF */
  printf("%s", buf); /* Process the input command */
}

int echo_line(int connfd) {
  ssize_t n;
```

```
  char buf[1];

  while ((n = Rio_readn(connfd, buf, 1)) > 0) {
    Rio_writen(connfd, buf, n);
    if (buf[0] = '\n')
      return 0;
  }
  return -1;
}
```

run server

```
(cd chapter12/code; make && ./12.22 5000)
```

input command in terminal or telnet to test server

```
telnet 127.0.0.1 5000
```

12.23

code in pic 12-8 save as file `12.23.bug.c`

run server

```
(cd chapter12/code; make && ./12.23.bug)
```

let's figure out when server will fail, see code `12.23.client.c`

```
/*
 * 12.23.client.c - An echo client
 */
#include "csapp.h"

int main(int argc, char **argv)
{
  int clientfd;
  char *host, *port;
  char *buf = "something to send\n";
  rio_t rio;

  host = "127.0.0.1";
  port = "5000";

  clientfd = Open_clientfd(host, port);

  Rio_readinitb(&rio, clientfd);
  Rio_writen(clientfd, buf, strlen(buf));
  /*Close(clientfd);*/
  exit(0);
}
```

line 20 `Close(clientfd)` are commented. when server is running, open another terminal and run `./12.23.client`, server will exit with error

```
Rio_readlineb error: Connection reset by peer
```

server can't read EOF because client doesn't call `Close`

how to fix:

just output error infomation and doesn't exit

```
--- 12.23.bug.c    2017-11-09 02:57:43.655936083 +0000
+++ 12.23.c    2017-11-09 02:57:43.655936083 +0000
@@ -1,7 +1,5 @@
 /*
- * 12.23.bug.c - A concurrent echo server based on select
- *
- * bug in this file
+ * 12.23.c - A concurrent echo server based on select
  */
 #include "csapp.h"

@@ -105,15 +103,21 @@
     /* If the descriptor is ready, echo a text line from it */
     if ((connfd > 0) && (FD_ISSET(connfd, &p->ready_set))) {
       p->nready--;
-      if ((n = Rio_readlineb(&rio, buf, MAXLINE)) != 0) {
+      if ((n = rio_readlineb(&rio, buf, MAXLINE)) > 0) {
         byte_cnt += n; //line:conc:echoservers:beginecho
         printf("Server received %d (%d total) bytes on fd %d\n"
,
             n, byte_cnt, connfd);
         Rio_writen(connfd, buf, n); //line:conc:echoservers:end
echo
       }
-
       /* EOF detected, remove descriptor from pool */
+      else if (n == 0) {
+        Close(connfd); //line:conc:echoservers:closeconnfd
+        FD_CLR(connfd, &p->read_set); //line:conc:echoservers:b
eginremove
+        p->clientfd[i] = -1;          //line:conc:echoservers:e
ndremove
+      }
+      /* n == -1, it's an error */
```

```
       else {
+         fprintf(stderr, "error in fd %d, close fd %d connection
\n", connfd, connfd);
         Close(connfd); //line:conc:echoservers:closeconnfd
         FD_CLR(connfd, &p->read_set); //line:conc:echoservers:b
eginremove
         p->clientfd[i] = -1;          //line:conc:echoservers:e
ndremove
```

12.24

if don't pass pointer param which points to same data block, functions

```
rio_readn
rio_writen
rio_readinitb
rio_readlineb
rio_readnb
```

are all implicitly reentrant functions

12.25

```c
/*
 * A thread-safe version of echo that counts the total number
 * of bytes received from clients.
 */
/* $begin echo_cnt */
#include "csapp.h"

static int byte_cnt;  /* Byte counter */
static sem_t mutex;   /* and the mutex that protects it */

static void init_echo_cnt(void)
{
    Sem_init(&mutex, 0, 1);
    byte_cnt = 0;
}

void echo_cnt(int connfd)
{
    int n;
    char buf[MAXLINE];
    rio_t rio;
    static pthread_once_t once = PTHREAD_ONCE_INIT;

    Pthread_once(&once, init_echo_cnt); //line:conc:pre:pthreado
nce
    Rio_readinitb(&rio, connfd);        //line:conc:pre:rioinitb
    while((n = Rio_readlineb(&rio, buf, MAXLINE)) != 0) {
    P(&mutex);
    byte_cnt += n; //line:conc:pre:cntaccess1
    printf("server received %d (%d total) bytes on fd %d\n",
           n, byte_cnt, connfd); //line:conc:pre:cntaccess2
    V(&mutex);
    Rio_writen(connfd, buf, n);
    }
}
/* $end echo_cnt */
```

thread safe?

Yes, mutex make it safe

reentrant?

No, share the same mutex

12.26

```
struct hostent *gethostbyname(const char *name)

struct hostent {
  char *h_name;
  char **h_aliases;
  int  h_addrtype;
  int  h_length;
  char **h_addr_list;
}
```

copy int, copy char, *copy char** in struct hostent are 3 different ways.

```
/*
 * 12.26.c
 */
#include <stdio.h>
#include "csapp.h"

/*
 * struct hostent *gethostbyname(const char *name)
 *
 * struct hostent {
 *   char *h_name;
 *   char **h_aliases;
 *   int  h_addrtype;
 *   int  h_length;
 *   char **h_addr_list;
 * }
 */
static sem_t mutex;

static void init_mutex(void) {
  Sem_init(&mutex, 0, 1);
}

struct hostent *gethostbyname_ts(const char *name, struct hosten
t *host) {
```

```
  struct hostent *sharehost;

  P(&mutex);
  sharehost = gethostbyname(name);
  // copy int
  host->h_addrtype = sharehost->h_addrtype;
  host->h_length = sharehost->h_length;
  // copy char *
  host->h_name = (char*)Malloc(strlen(sharehost->h_name));
  strcpy(host->h_name, sharehost->h_name);
  // copy char **
  int i;
  for (i = 0; sharehost->h_aliases[i] != NULL; i++) {}
  host->h_aliases = (char**)Malloc(sizeof(char*) * (i+1));
  for (i = 0; sharehost->h_aliases[i] != NULL; i++) {
    // copy every char *
    host->h_aliases[i] = (char*)Malloc(strlen(sharehost->h_alias
es[i]));
    strcpy(host->h_aliases[i], sharehost->h_aliases[i]);
  }
  host->h_aliases[i] = NULL;

  for (i = 0; sharehost->h_addr_list[i] != NULL; i++) {}
  host->h_addr_list = (char**)Malloc(sizeof(char*) * (i+1));
  for (i = 0; sharehost->h_addr_list[i] != NULL; i++) {
    // copy every char *
    host->h_addr_list[i] = (char*)Malloc(strlen(sharehost->h_add
r_list[i]));
    strcpy(host->h_addr_list[i], sharehost->h_addr_list[i]);
  }
  host->h_addr_list[i] = NULL;
  V(&mutex);

  return host;
}

int main(int argc, char* argv[]) {
  init_mutex();

  struct hostent host;
```

```
    gethostbyname_ts("127.0.0.1", &host);
    // result in &host

    return 0;
}
```

12.27

```c
FILE *fpin, *fpout;
fpin = fdopen(sockfd, "r");
fpout = fdopen(sockfd, "w");

// read and write

fclose(fpin);
fclose(fpout);
```

ref: 10.11

`fdopen` open 2 stream on same sockfd, `fdclose` will close sockfd under stream. if you call `fclose` 2 stream on the same sockfd, the second `fclose` will fail.

image one thread execute code and open 2 stream on fd N. after execution of line `fclose(fpin);` , program create another thread and execute the same code.

but

after `fclose(fpin);` in thread 1, fd N is reusable again. assume thread 2 use the fd N again: thread 1 execute line `fclose(fpout);` close the fd that thread 2 is using. it'll cause something unpredicted.

12.28

No effect on deadlock

1

thread 1	thread 2
P(s)	P(s)
P(t)	P(t)
V(s)	V(s)
V(t)	V(t)

```
    +
    |
    |
    +                   +--------------------+
V(t)|                   |                    |
    |                   |                    |
    +                   |  unsafe region t   |
    |                   |                    |
    |                   |                    |
    +         +---------+---------+          |
V(s)|         |         |         |          |
    |         |         |         |          |
    +         |         |         |          |
    |         |         |         |          |
    |         |         |         |          |
    +         |         +---------+----------+
P(t)|         |                   |
    |         |  unsafe region s  |
    +         |                   |
    |         |                   |
    |         |                   |
    +         +-------------------+
P(s)|
    |
    +
    |
    |
    +----+----+----+----+----+----+----+----+----+  thread 1
          P(s)      P(t)      V(s)      V(t)
```

2

thread 1	thread 2
P(s)	P(s)
P(t)	P(t)
V(s)	V(t)
V(t)	V(s)

```
    +
    |
    |
    +         +-------------------+
V(s)|         |                   |
    |         |                   |
    +         |                   |
    |         |                   |
    |         |                   |
    +         |         +---------+----------+
V(t)|         |         |         |          |
    |         |         |         | unsafe   |
    +         |         |         | region   |
    |         |         |         |    t     |
    |         |         |         |          |
    +         |         +---------+----------+
P(t)|         |                   |
    |         |                   |
    +         | unsafe region s   |
    |         |                   |
    |         |                   |
    +         +-------------------+
P(s)|
    |
    +
    |
    |
    +----+----+----+----+----+----+----+----+----+  thread 1
          P(s)      P(t)      V(s)      V(t)
```

3

thread 1	thread 2
P(s)	P(s)
P(t)	P(t)
V(t)	V(s)
V(s)	V(t)

```
    +
    |
    |
    +                   +---------+
V(t)|                   | unsafe  |
    |                   | region  |
    +                   |   t     |
    |                   |         |
    |                   |         |
    +         +---------+---------+----------+
V(s)|         |         |         |          |
    |         |         |         |          |
    +         |         |         |          |
    |         |         |         |          |
    |         |         |         |          |
    +         |         +---------+          |
P(t)|         |                              |
    |         |                              |
    +         |    unsafe region s           |
    |         |                              |
    |         |                              |
    +         +------------------------------+
P(s)|
    |
    +
    |
    |
    +----+----+----+----+----+----+----+----+----+  thread 1
          P(s)      P(t)      V(t)      V(s)
```

4

thread 1	thread 2
P(s)	P(s)
P(t)	P(t)
V(t)	V(t)
V(s)	V(s)

```
    +
    |
    |
    +         +------------------------------+
V(s)|         |                              |
    |         |                              |
    +         |                              |
    |         |                              |
    |         |                              |
    +         |         +----------+         |
V(t)|         |         | unsafe   |         |
    |         |         | region   |         |
    +         |         |    t     |         |
    |         |         |          |         |
    |         |         |          |         |
    +         |         +----------+         |
P(t)|         |                              |
    |         |                              |
    +         |    unsafe region s           |
    |         |                              |
    |         |                              |
    +         +------------------------------+
P(s)|
    |
    +
    |
    |
    +----+----+----+----+----+----+----+----+----+  thread 1
          P(s)      P(t)      V(t)      V(s)
```

12.29

no deadlock

initial: a = 1, b = 1, c = 1

thread 1	thread 2
P(a)	P(c)
P(b)	P(b)
V(b)	V(b)
P(c)	V(c)
V(c)	----
V(a)	----

thread 2 doesn't manipulate mutex a and initial a is 1, so P(a), V(a) don't affect deadlock status.

thread 1	thread 2
P(b)	P(c)
V(b)	P(b)
P(c)	V(b)
V(c)	V(c)

```
    +
    |
    |
    +                             +----------+
V(c)|                             |          |
    |                             |          |
    +                             |          |
    |                             |          |
    |                             |          |
    +         +----------+        |          |
V(b)|         |          |        |          |
    |         |          |        |          |
    +         |          |        |          |
    |         |          |        |          |
    |         |          |        |          |
    +         +----------+        |          |
P(b)|                             |          |
    |                             |          |
    +                             |          |
    |                             |          |
    |                             |          |
    +                             +----------+
P(c)|
    |
    +
    |
    |
    +----+----+----+----+----+----+----+----+----+  thread 1
          P(b)      V(b)      P(c)      V(c)
```

12.30

initial: a = 1, b = 1, c = 1

thread 1	thread 2	thread 3
P(a)	P(c)	P(c)
P(b)	P(b)	V(c)
V(b)	V(b)	P(b)
P(c)	V(c)	P(a)
V(c)	P(a)	V(a)
V(a)	V(a)	V(b)

A.

thread 1: a&b, a&c

thread 2: b&c

thread 3: a&b

B.

thread 2 and thread 3

C.

keep same order P(a), P(b), P(c) in every thread

12.31

```c
/*
 * 12.31.c
 */
#include <stdio.h>
#include "csapp.h"

sigjmp_buf buf;

void sigchild_handler(int sig) {
  siglongjmp(buf, 1);
}

char *tfgets(char *s, int size, FILE *stream) {
  if (Fork() == 0) {
    Sleep(5);
    exit(0);
  }

  switch (sigsetjmp(buf, 1)) {
    case 0:
      Signal(SIGCHLD, sigchild_handler);
      return fgets(s, size, stream);
    case 1:
      return NULL;
  }
}

int main(int argc, char* argv[]) {
  char buf[MAXLINE];

  if (tfgets(buf, MAXLINE, stdin) == NULL)
    printf("BOOM!\n");
  else
    printf("%s", buf);

  return 0;
}
```

12.32

key is last param of `select`

```
/*
 * 12.32.c
 */
#include <stdio.h>
#include "csapp.h"

char *tfgets(char *s, int size, FILE *stream) {
  fd_set read_set;
  FD_ZERO(&read_set);
  FD_SET(STDIN_FILENO, &read_set);

  struct timeval timeout;
  timeout.tv_sec = 5;
  timeout.tv_usec = 0;

  Select(1, &read_set, NULL, NULL, &timeout);
  if (FD_ISSET(STDIN_FILENO, &read_set))
    return fgets(s, size, stream);
  else
    return NULL;
}

int main(int argc, char* argv[]) {
  char buf[MAXLINE];

  if (tfgets(buf, MAXLINE, stdin) == NULL)
    printf("BOOM!\n");
  else
    printf("%s", buf);

  return 0;
}
```

12.33

```c
/*
 * 12.33.c
 */
#include <stdio.h>
#include "csapp.h"

struct pack {
  char *s;
  int size;
  FILE *stream;
};

char *ptr = NULL;
int timeout = -1;

void *thread_read(void *vargp) {
  struct pack p = *(struct pack *)vargp;
  ptr = fgets(p.s, p.size, p.stream);
  timeout = 0;
}

void *thread_sleep(void *vargp) {
  Sleep(5);
  timeout = 1;
}

char *tfgets(char *s, int size, FILE *stream) {
  pthread_t tid_read;
  pthread_t tid_sleep;
  struct pack p;

  p.s = s;
  p.size = size;
  p.stream = stream;
  Pthread_create(&tid_read, NULL, thread_read, (void*)&p);

  Pthread_create(&tid_sleep, NULL, thread_sleep, NULL);
```

```
  // wait 2 thread race result
  while(timeout == -1) {}

  if (timeout == 1) {
    Pthread_cancel(tid_read);
    return NULL;
  } else {
    Pthread_cancel(tid_sleep);
    return ptr;
  }
}

int main(int argc, char* argv[]) {
  char buf[MAXLINE];

  if (tfgets(buf, MAXLINE, stdin) == NULL)
    printf("BOOM!\n");
  else
    printf("%s", buf);

  return 0;
}
```

12.34

matrix size and thread number

```
/*
 * 12.34.h
 */

#define N 640
#define M 640

#define THREAD (1<<4)
#define ROWS_PER_THREAD (N / THREAD)
```

non concurrent version

```c
/*
 * 12.34.non.concurrent.c
 */
#include <stdio.h>
#include "csapp.h"
#include "12.34.h"

int M1[N][M];
int M2[N][M];

int MUL12[N][M];

void non_concurrent_mul(void) {
  int i, j, k;
  for (i = 0; i < N; i++)
    for (j = 0; j < N; j++) {
      int sum = 0;
      for (k = 0; k < M; k++) {
        sum += M1[i][k] * M2[k][j];
      }
      MUL12[i][j] = sum;
    }
}

int main(int argc, char* argv[]) {
  non_concurrent_mul();
  return 0;
}
```

concurrent version

```c
/*
 * 12.34.concurrent.c
 */
#include <stdio.h>
#include "csapp.h"
#include "12.34.h"

int M1[N][M];
```

```
int M2[N][M];

int MUL12[N][M];

void *thread_mul(void *vargp) {
  int idx = *(int*)vargp;
  int start = ROWS_PER_THREAD * idx;
  int i, j, k;
  for (i = start; i < start+ROWS_PER_THREAD; i++)
    for (j = 0; j < N; j++) {
      int sum = 0;
      for (k = 0; k < M; k++) {
        sum += M1[i][k] * M2[k][j];
      }
      MUL12[i][j] = sum;
    }
}

void concurrent_mul(void) {
  pthread_t tid[THREAD];
  int param[THREAD];
  int i;

  for (i = 0; i < THREAD; i++) {
    param[i] = i;
    Pthread_create(&tid[i], NULL, thread_mul, &param[i]);
  }
  for (i = 0; i < THREAD; i++) {
    Pthread_join(tid[i], NULL);
  }
}

int main(int argc, char* argv[]) {
  concurrent_mul();
  return 0;
}
```

measure performance

```
(cd chapter12/code; make clean && make && make measure)
```

output

```
(time ./12.34.non.concurrent)
0.90user 0.00system 0:00.90elapsed 99%CPU (0avgtext+0avgdata 370
4maxresident)k
0inputs+0outputs (0major+756minor)pagefaults 0swaps

// single cpu run time 0.9s

(time ./12.34.concurrent)
2.20user 0.00system 0:00.64elapsed 341%CPU (0avgtext+0avgdata 38
96maxresident)k
0inputs+0outputs (0major+1462minor)pagefaults 0swaps

// 4 cpu, total run time 2.2s(every cpu run 0.55s, concurrent!!)
```

more detialed

thread(t)	1	2	4	8	16
core(p)	1	2	4	4	4
time(Tp)	0.86	0.466	0.626	0.627	0.628
speedup(Sp)	1	1.84	1.37	1.37	1.37
efficiency(Ep)	100%	92.2%	34.3%	34.3%	34.3%

12.35

see origin tiny server in section 11.6 on book

the key is `close(connfd)` in both parent and child process to ensure close fd and reuse it or it'll failed: `Accept error: too many open files`

```
--- 12.tiny.c    2017-11-09 02:57:43.655936083 +0000
+++ 12.35.c    2017-11-09 02:57:43.655936083 +0000
@@ -1,6 +1,8 @@
 /*
- * tiny.c - A simple, iterative HTTP/1.0 Web server that uses t
he
+ * 12.35.c - A simple, iterative HTTP/1.0 Web server that uses
the
  *     GET method to serve static and dynamic content.
+ *
+ *     concurrent server in multi process way.
  */
 #include "csapp.h"

@@ -35,8 +37,14 @@
     Getnameinfo((SA *) &clientaddr, clientlen, hostname, MAXLIN
E,
         port, MAXLINE, 0);
     printf("Accepted connection from (%s, %s)\n", hostname, por
t);
-    doit(connfd);                                              /
/line:netp:tiny:doit
-    Close(connfd);                                             /
/line:netp:tiny:close
+
+    if (Fork() == 0) {
+      Close(listenfd);
+      doit(connfd);
 //line:netp:tiny:doit
+      Close(connfd);
 //line:netp:tiny:close
+      exit(0);
+    }
+    Close(connfd);
   }
 }
```

run server

```
(cd chapter12/code; make && ./12.35)
```

open another terminal and benchmark it

```
wrk -d4 http://localhost:5000
```

output

```
Running 4s test @ http://localhost:5000
  2 threads and 10 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency    62.20ms   87.89ms 348.01ms   82.24%
    Req/Sec   397.69    184.95   767.00     58.33%
  2585 requests in 4.00s, 578.09KB read
Requests/sec:    645.64
Transfer/sec:    144.39KB
```

12.36

check section 11.6 and 12.2.1, combine code pieces together

code directory: chapter12/code/12.36

origin tiny server: tiny.h tiny.c

select echoserver: echoserver.h echoserver.c

key points:

- move main loop to main.c file
- `check_clients` in echoserver call `doit` (from tiny server) and close fd

benchmark it

```
Running 4s test @ http://localhost:5000
  2 threads and 10 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency    62.72ms   91.68ms 368.55ms   82.94%
    Req/Sec   615.27    398.96     1.46k    65.91%
  3649 requests in 4.00s, 816.04KB read
Requests/sec:    911.45
Transfer/sec:    203.83KB
```

12.37

see origin tiny server in section 11.6 on book

the key is how to pass connfd into thread

```
--- 12.tiny.c    2017-11-09 02:57:43.655936083 +0000
+++ 12.37.c    2017-11-09 02:57:43.655936083 +0000
@@ -13,9 +13,13 @@
 void clienterror(int fd, char *cause, char *errnum,
     char *shortmsg, char *longmsg);

+void *thread(void *vargp);
+
 int main(int argc, char **argv)
 {
   int listenfd, connfd;
+  int *connfdp;
+  pthread_t tid;
   char hostname[MAXLINE], port[MAXLINE];
   socklen_t clientlen;
   struct sockaddr_storage clientaddr;
@@ -35,11 +39,23 @@
     Getnameinfo((SA *) &clientaddr, clientlen, hostname, MAXLIN
E,
         port, MAXLINE, 0);
     printf("Accepted connection from (%s, %s)\n", hostname, por
t);
-    doit(connfd);                                              /
/line:netp:tiny:doit
-    Close(connfd);                                             /
/line:netp:tiny:close
+
+    connfdp = (int*)Malloc(sizeof(int));
+    *connfdp = connfd;
+    Pthread_create(&tid, NULL, thread, connfdp);
   }
 }

+void *thread(void *vargp) {
```

```
+   int connfd = *(int*)vargp;
+   Pthread_detach(Pthread_self());
+   Free(vargp);
+
+   doit(connfd);
+   Close(connfd);
+   return NULL;
+}
+
 /*
  * doit - handle one HTTP request/response transaction
  */
```

run server

```
(cd chapter12/code; make && ./12.37)
```

open another terminal and benchmark it

```
wrk -d4 http://localhost:5000
```

output

```
Running 4s test @ http://localhost:5000
  2 threads and 10 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency    60.24ms   87.59ms 354.24ms   82.32%
    Req/Sec   481.88    276.48     1.09k    66.67%
  3151 requests in 4.10s, 704.67KB read
Requests/sec:    768.63
Transfer/sec:    171.89KB
```

12.38

code in directory `chapter12/code/12.38`

files:

- sbuf.h, sbuf.c: prethreading package from section 12.5.4 in book. differences are that add new functions `sbuf_full` , `sbuf_empty` .
- tiny.h, tiny.c: origin tiny server from section 11.6 in book. differences are we seperate function declarations into a .h file
- main.c: listen new connections, create threads and adjust threads number dynamicly

see new functions in sbuf, `sbuf_full` and `sbuf_empty`

```
/* Empty buf? */
int sbuf_empty(sbuf_t *sp) {
  int e;
  P(&sp->mutex);                          /* Lock the buffer */
  e = sp->front == sp->rear;
  V(&sp->mutex);                          /* Lock the buffer */
  return e;
}

/* Full buf? */
int sbuf_full(sbuf_t *sp) {
  int f;
  P(&sp->mutex);                          /* Lock the buffer */
  f = (sp->rear - sp->front) == sp->n;
  V(&sp->mutex);                          /* Lock the buffer */
  return f;
}
```

main.c

```
/*
 * main.c
 */
#include <stdio.h>
#include "../csapp.h"
```

```
#include "tiny.h"
#include "sbuf.h"

#define SBUFSIZE  4
#define INIT_THREAD_N  1
#define THREAD_LIMIT 4096

static int nthreads;
static sbuf_t sbuf; /* Shared buffer of connected descriptors */

// thread info
typedef struct {
  pthread_t tid;
  sem_t mutex;
} ithread;

static ithread threads[THREAD_LIMIT];

// init work
void init(void);
// function for create server thread
void *serve_thread(void *vargp);
/*
 * creating thread that adjust total thread count according to s
buf situation
 *
 * if sbuf is empty, double threads
 * if sbuf is full, half threads
 */
void *adjust_threads(void *);
// from start to end, create (end - start) new server threads
void create_threads(int start, int end);

int main(int argc, char **argv) {
  int i, listenfd, connfd;
  socklen_t clientlen;
  struct sockaddr_storage clientaddr;
  pthread_t tid;

  if (argc != 2) {
```

```
    fprintf(stderr, "usage: %s <port>\n", argv[0]);
    fprintf(stderr, "use default port 5000\n");
    listenfd = Open_listenfd("5000");
  } else {
    listenfd = Open_listenfd(argv[1]);
  }

  init();

  Pthread_create(&tid, NULL, adjust_threads, NULL);

  while (1) {
    clientlen = sizeof(struct sockaddr_storage);
    connfd = Accept(listenfd, (SA *) &clientaddr, &clientlen);
    sbuf_insert(&sbuf, connfd); /* Insert connfd in buffer */
  }
}

void init(void) {
  nthreads = INIT_THREAD_N;
  sbuf_init(&sbuf, SBUFSIZE);

  // create initail server threads
  create_threads(0, nthreads);
}

void *serve_thread(void *vargp) {
  int idx = *(int*)vargp;
  Free(vargp);

  while (1) {
    // get lock first
    // thread can't be kill now
    P(&(threads[idx].mutex));

    int connfd = sbuf_remove(&sbuf);
    doit(connfd);
    Close(connfd);

    // service ends and release lock
```

```
    // so thread can be kill at this time
    V(&(threads[idx].mutex));
  }
}

void create_threads(int start, int end) {
  int i;
  for (i = start; i < end; i++) {
    // init mutex for every new thread
    Sem_init(&(threads[i].mutex), 0, 1);
    // create thread
    int *arg = (int*)Malloc(sizeof(int));
    *arg = i;
    // pass thread index in array into thread inside
    Pthread_create(&(threads[i].tid), NULL, serve_thread, arg);
  }
}

void *adjust_threads(void *vargp) {
  sbuf_t *sp = &sbuf;

  while (1) {
    // if sbuf is full, double threads
    if (sbuf_full(sp)) {
      if (nthreads == THREAD_LIMIT) {
        fprintf(stderr, "too many threads, can't double\n");
        continue;
      }

      // double n
      int dn = 2 * nthreads;
      create_threads(nthreads, dn);
      nthreads = dn;
      continue;
    }

    // half threads
    if (sbuf_empty(sp)) {
      if (nthreads == 1)
        continue;
```

```
        // half n
        int hn = nthreads / 2;
        /*
         * all server thread are divide to 2 parts
         *
         * keep [0, hn] running
         * kill [hn, nthreads] threads
         *
         * if you want to kill a thread, you must get the lock bef
ore it so you
         * won't kill a thread which is offering service.
         */
        int i;
        for (i = hn; i < nthreads; i++) {
          P(&(threads[i].mutex));
          Pthread_cancel(threads[i].tid);
          V(&(threads[i].mutex));
        }
        nthreads = hn;
        continue;
      }
    }
}
```

run server

```
(cd chapter12/code/12.38; make && ./main)
```

open another terminal and benchmark it

```
wrk -d4 http://localhost:5000
```

output

```
Running 4s test @ http://localhost:5000
  2 threads and 10 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency   130.44us  183.40us   9.03ms   98.06%
    Req/Sec    11.08k     5.09k   17.78k    53.66%
  90380 requests in 4.10s, 19.74MB read
Requests/sec:  22039.72
Transfer/sec:      4.81MB
```

12.39

```
+------+      +------+       +------+
|      |----->|      |------>|      |
|client|      |proxy |       |server|
|      |<-----|      |<------|      |
+------+      +------+       +------+
      as server      as client
```

proxy play with client as a server, with server as a client, interesting.

the key point is when normal situation

```
+------+      +------+
|      |----->|      |
|client|      |server|
|      |<-----|      |
+------+      +------+
```

HTTP request from client is something like

```
GET / HTTP/1.1
Host: address:port
```

but with proxy, it'll be

```
GET http://address:port/ HTTP/1.1
Host: address:port
```

so proxy have to separate host from uri

A.

```
/*
 * proxy.c
 *
 * visited url log to file log.list
```

```
 * block url base on entry from file block.list
 */
#include <stdio.h>
#include "../csapp.h"

// block.list limit entry num
#define MAXENTRY 100

int separate_uri(char *uri, char *host, char *port, char *path);
void parse_block_file(char *filename, char list[MAXENTRY][MAXLIN
E], int limit);
int blocked_uri(char *uri, char list[MAXENTRY][MAXLINE]);

int main(int argc, char **argv) {
  int i, listenfd, connfd;
  int clientfd;

  socklen_t clientlen;
  struct sockaddr_storage clientaddr;

  rio_t client_rio, server_rio;
  char c_buf[MAXLINE], s_buf[MAXLINE];
  ssize_t sn, cn;

  char method[MAXLINE], uri[MAXLINE], version[MAXLINE];
  char host[MAXLINE], port[MAXLINE], path[MAXLINE];

  char block_list[MAXENTRY][MAXLINE];
  int logfd;
  char log_buf[MAXLINE];

  if (argc != 2) {
    fprintf(stderr, "usage: %s <port>\n", argv[0]);
    fprintf(stderr, "use default port 5000\n");
    listenfd = Open_listenfd("5000");
  } else {
    listenfd = Open_listenfd(argv[1]);
  }

  logfd = Open("log.list", O_WRONLY | O_APPEND, 0);
```

```
  memset(block_list, '\0', MAXLINE * MAXENTRY);
  parse_block_file("block.list", block_list, MAXENTRY);

  while (1) {
    // wait for connection as a server
    clientlen = sizeof(struct sockaddr_storage);
    connfd = Accept(listenfd, (SA *) &clientaddr, &clientlen);
    Rio_readinitb(&server_rio, connfd);

    /*
     * if uri is full path url like http://localhost:8000/server
.c
     * remove host part http://localhost:8000
     * only pass /server.c to server
     */
    // parse HTTP request first line
    if (!Rio_readlineb(&server_rio, s_buf, MAXLINE)) {
      Close(connfd);
      continue;
    }
    sscanf(s_buf, "%s %s %s", method, uri, version);
    // if uri is blocked?
    if (blocked_uri(uri, block_list)) {
      printf("%s is blocked\n", uri);
      Close(connfd);
      continue;
    }
    // log visit
    sprintf(log_buf, "visit url: %s\n", uri);
    Write(logfd, log_buf, strlen(log_buf));

    memset(host, '\0', MAXLINE);
    memset(port, '\0', MAXLINE);
    memset(path, '\0', MAXLINE);
    int res;
    if ((res = separate_uri(uri, host, port, path)) == -1) {
      fprintf(stderr, "not http protocol\n");
      Close(connfd);
      continue;
```

```
    } else if (res == 0) {
      fprintf(stderr, "not a abslute request path\n");
      Close(connfd);
      continue;
    }

    // connect server as a client
    clientfd = Open_clientfd(host, port);
    Rio_readinitb(&client_rio, clientfd);

    /*
     *  browser  -->  proxy  -->  server
     *
     *  send requests
     */
    // write first request line
    sprintf(s_buf, "%s %s %s\n", method, path, version);
    Rio_writen(clientfd, s_buf, strlen(s_buf));
    printf("%s", s_buf);
    do {
      // pass next http requests
      sn = Rio_readlineb(&server_rio, s_buf, MAXLINE);
      printf("%s", s_buf);
      Rio_writen(clientfd, s_buf, sn);
    } while(strcmp(s_buf, "\r\n"));

    /*
     *  server  -->  proxy  -->  browser
     *
     *  server send response back
     */
    while ((cn = Rio_readlineb(&client_rio, c_buf, MAXLINE)) != 0
)
      Rio_writen(connfd, c_buf, cn);

    Close(connfd);
    Close(clientfd);
  }

  Close(logfd);
```

```
}

/*
 * if uri is abslute path url like
 *   http://localhost:8888/something
 *   or
 *   http://localhost/something (port default is 80)
 * separate into three part and return 1
 *
 * if uri is relative path like /something
 * do nothing and return 0
 *
 * if uri is abslute path and not http protocal like https/ftp/e
tc
 * do nothing, return -1, it's error
 */
int separate_uri(char *uri, char *host, char *port, char *path)
{
  // relative path
  if (uri[0] == '/')
    return 0;

  // abslute path
  char *prefix = "http://";
  int prelen = strlen(prefix);
  // if not http protocal, error
  if (strncmp(uri, prefix, prelen) != 0)
    return -1;

  char *start, *end;
  start = uri + prelen;
  end = start;

  // copy host
  while (*end != ':' && *end != '/') {
    end++;
  }
  strncpy(host, start, end-start);

  // port is provided
```

```
  if (*end == ':') {
    // skip ':'
    ++end;
    start = end;
    // copy port
    while (*end != '/')
      end++;
    strncpy(port, start, end-start);
  } else {
    // port is not provided, defualt 80
    strncpy(port, "80", 2);
  }

  // copy path
  strcpy(path, end);
}

/*
 * read block file, parse all the entries and save into list
 * entries count no more than limit
 */
void parse_block_file(char *filename, char list[MAXENTRY][MAXLIN
E], int limit) {
  int blockfd;
  char block_buf[MAXLINE];
  rio_t block_rio;
  ssize_t block_n;

  blockfd = Open(filename, O_RDONLY, 0);
  Rio_readinitb(&block_rio, blockfd);

  memset(block_buf, '\0', MAXLINE);
  int num = 0;
  while ((block_n = Rio_readlineb(&block_rio, block_buf, MAXLINE
)) != 0) {
    // exceed limit
    if (num == limit)
      break;

    // right entry
```

```
    if (strncmp(block_buf, "http://", 7) == 0) {
      strcpy(list[num], block_buf);
      num++;
    }

    // comment or not right format entry
    // do nothing
  }

  Close(blockfd);
}

/*
 * if uri is in list, return true
 * if not, return false
 */
int blocked_uri(char *uri, char list[MAXENTRY][MAXLINE]) {
  int i;
  for (i = 0; list[i][0] != '\0'; i++)
    if (strncmp(uri, list[i], strlen(uri)) == 0)
      return 1;

  return 0;
}
```

B.

move process into thread function

```
/*
 * proxy-thread.c  multi thread deal with concurrent
 *
 * visited url log to file log.list
 * block url base on entry from file block.list
 */
#include <stdio.h>
#include "../csapp.h"

// block.list limit entry num
```

```
#define MAXENTRY 100

int separate_uri(char *uri, char *host, char *port, char *path);
void parse_block_file(char *filename, char list[MAXENTRY][MAXLIN
E], int limit);
int blocked_uri(char *uri, char list[MAXENTRY][MAXLINE]);
void *proxy_thread(void *vargp);

// url black list
static char block_list[MAXENTRY][MAXLINE];
// log file fd
static int logfd;

int main(int argc, char **argv) {
  int listenfd;
  socklen_t clientlen;
  struct sockaddr_storage clientaddr;
  int *connfdp;
  pthread_t tid;

  if (argc != 2) {
    fprintf(stderr, "usage: %s <port>\n", argv[0]);
    fprintf(stderr, "use default port 5000\n");
    listenfd = Open_listenfd("5000");
  } else {
    listenfd = Open_listenfd(argv[1]);
  }

  logfd = Open("log.list", O_WRONLY | O_APPEND, 0);
  memset(block_list, '\0', MAXLINE * MAXENTRY);
  parse_block_file("block.list", block_list, MAXENTRY);

  while (1) {
    // wait for connection as a server
    clientlen = sizeof(struct sockaddr_storage);
    connfdp = Malloc(sizeof(int));
    *connfdp = Accept(listenfd, (SA *) &clientaddr, &clientlen);
    // new thread
    Pthread_create(&tid, NULL, proxy_thread, connfdp);
  }
```

```
  Close(logfd);
}

void *proxy_thread(void *vargp) {
  pthread_t tid = Pthread_self();
  Pthread_detach(tid);

  int connfd = *(int*)vargp;
  Free(vargp);

  rio_t client_rio, server_rio;
  char c_buf[MAXLINE], s_buf[MAXLINE];
  ssize_t sn, cn;

  char method[MAXLINE], uri[MAXLINE], version[MAXLINE];
  char host[MAXLINE], port[MAXLINE], path[MAXLINE];

  char log_buf[MAXLINE];

  int clientfd;
  Rio_readinitb(&server_rio, connfd);
  /*
   * if uri is full path url like http://localhost:8000/server.c
   * remove host part http://localhost:8000
   * only pass /server.c to server
   */
  // parse HTTP request first line
  if (!Rio_readlineb(&server_rio, s_buf, MAXLINE)) {
    Close(connfd);
    return NULL;
  }
  sscanf(s_buf, "%s %s %s", method, uri, version);
  // if uri is blocked?
  if (blocked_uri(uri, block_list)) {
    printf("thread %ld: %s is blocked\n", tid, uri);
    Close(connfd);
    return NULL;
  }
  // log visit
```

```
  sprintf(log_buf, "thread %ld: visit url: %s\n", tid, uri);
  Write(logfd, log_buf, strlen(log_buf));

  memset(host, '\0', MAXLINE);
  memset(port, '\0', MAXLINE);
  memset(path, '\0', MAXLINE);
  int res;
  if ((res = separate_uri(uri, host, port, path)) == -1) {
    fprintf(stderr, "tid %ld: not http protocol\n", tid);
    Close(connfd);
    return NULL;
  } else if (res == 0) {
    fprintf(stderr, "tid %ld: not a abslute request path\n", tid
);
    Close(connfd);
    return NULL;
  }

  // connect server as a client
  clientfd = Open_clientfd(host, port);
  Rio_readinitb(&client_rio, clientfd);

  /*
   *  browser  -->  proxy  -->  server
   *
   *  send requests
   */
  // write first request line
  sprintf(s_buf, "%s %s %s\n", method, path, version);
  Rio_writen(clientfd, s_buf, strlen(s_buf));
  printf("tid %ld: %s", tid, s_buf);
  do {
    // pass next http requests
    sn = Rio_readlineb(&server_rio, s_buf, MAXLINE);
    printf("tid %ld: %s", tid, s_buf);
    Rio_writen(clientfd, s_buf, sn);
  } while(strcmp(s_buf, "\r\n"));

  /*
   *  server  -->  proxy  -->  browser
```

```
    *
    *  server send response back
    */
  while ((cn = Rio_readlineb(&client_rio, c_buf, MAXLINE)) != 0)
    Rio_writen(connfd, c_buf, cn);

  Close(connfd);
  Close(clientfd);
}

/*
 * if uri is abslute path url like
 *   http://localhost:8888/something
 *   or
 *   http://localhost/something (port default is 80)
 * separate into three part and return 1
 *
 * if uri is relative path like /something
 * do nothing and return 0
 *
 * if uri is abslute path and not http protocal like https/ftp/e
tc
 * do nothing, return -1, it's error
 */
int separate_uri(char *uri, char *host, char *port, char *path)
{
  // relative path
  if (uri[0] == '/')
    return 0;

  // abslute path
  char *prefix = "http://";
  int prelen = strlen(prefix);
  // if not http protocal, error
  if (strncmp(uri, prefix, prelen) != 0)
    return -1;

  char *start, *end;
  start = uri + prelen;
  end = start;
```

```
  // copy host
  while (*end != ':' && *end != '/') {
    end++;
  }
  strncpy(host, start, end-start);

  // port is provided
  if (*end == ':') {
    // skip ':'
    ++end;
    start = end;
    // copy port
    while (*end != '/')
      end++;
    strncpy(port, start, end-start);
  } else {
    // port is not provided, defualt 80
    strncpy(port, "80", 2);
  }

  // copy path
  strcpy(path, end);
}

void parse_block_file(char *filename, char list[MAXENTRY][MAXLIN
E], int limit) {
  int blockfd;
  char block_buf[MAXLINE];
  rio_t block_rio;
  ssize_t block_n;

  blockfd = Open(filename, O_RDONLY, 0);
  Rio_readinitb(&block_rio, blockfd);

  memset(block_buf, '\0', MAXLINE);
  int num = 0;
  while ((block_n = Rio_readlineb(&block_rio, block_buf, MAXLINE
)) != 0) {
    // exceed limit
```

```
    if (num == limit)
      break;

    // right entry
    if (strncmp(block_buf, "http://", 7) == 0) {
      strcpy(list[num], block_buf);
      num++;
    }

    // comment or not right format entry
    // do nothing
  }

  Close(blockfd);
}

/*
 * if uri is in list, return true
 * if not, return false
 */
int blocked_uri(char *uri, char list[MAXENTRY][MAXLINE]) {
  int i;
  for (i = 0; list[i][0] != '\0'; i++)
    if (strncmp(uri, list[i], strlen(uri)) == 0)
      return 1;

  return 0;
}
```

how to benchmark:

run proxy

```
(cd chapter12/code/12.39; make && ./proxy)
# or
(cd chapter12/code/12.39; make && ./proxy-thread)
```

open another terminal, start a local http server

```
(python3 -m http.server 8000)
```

open one another terminal, benchmark

```
# -X means proxy, -n means how many requests, -c means concurren
t
(ab -X localhost:5000 -n 100 -c 10 http://localhost:8000/)
```