系统级编程 - 选择题

Somnia1337

Week 01

Compared to a sequence of machine code instructions, a fragment of C code **may describe the same algorithm**.

Which of the following is able to describe a computation at the highest level of abstraction? **C++ code**

When using a debugger to find the cause of a program's incorrect behavior, **it is often necessary to start the program multiple times under the debugger**.

In visual C++, a Win32 console application is **the simplest type of application visual C+ + can generate**.

Consider the following fragment of C++ source code.

```
string msg;
unsigned int x;
int y;
cin >> msg >> x >> y;
cout << x + y;
```

复制

Which of the following is (are) true regarding execution of the segment?

- The input statement will always take the same amount of time to execute.
- The output statement will always be executed immediately after the input statement.
- If `x` and `y` are both positive, an integer greater than both will be printed.

none

Which of the following visual C++ objects are contained within a "project"?

- Files
- Visual C++ solutions
- Flow charts

I only

Which of the following does a debugger do?

- Analyze the source code to find programming errors.
- Decode machine code generated by a compiler.
- Stop execution of a program.

II and III only

Integrated programming environments make it difficult to mix and match tools from different sources. This is bad, because no single vendor is likely to be the source of all the best tools.

The correct sequence of GCC compilation process is preprocessing → compilation → assemble → linking.

The preprocessor removes the comments from the source code.

Week 02

In C, using default floating point settings, what happens when a floating-point computation results in an overflow? A special value "infinity" is computed, testable with `_finite()`.

Which of the following numerical operations is most likely to lead to loss of precision? floating-point addition (NOT multiplication)

Which of the following statements about floating-point numbers in C is true?

- Floating-point numbers are often only approximations of real numbers.
- A 32-bit float only approximates decimal fractions, but a 64-bit double represents them exactly.
- Floating-point numbers can represent any rational real number but not irrationals.

I only

What is the purpose of the exponent in floating point numbers? To indicate where the decimal or binary point should be.

What happens in a C program when an addition would cause integer overflow? An incorrect result is produced and execution continues.

In a computer with 4-byte words, which of the following C expressions tests whether `ptr` contains the address of a word?

- `(ptr & 3) == 0`
- `(ptr | 3) == 0`
- `(ptr % 4) == 0`

I and III only

How is `-10` (decimal) represented in an 8-bit 2's complement binary format? `11110110`

How is `46` (decimal) represented in an 8-bit 2's complement binary format? `101110`

What is the value of the C expression `0x1234 & 0x5432`? `0x1030`

What is the value of the C expression `0x1234 ^ 0x5432`? `0x4606`

In C, what is the binary number `11010101` in hexadecimal? `0xD5`

Which of the following could be represented by one bit of information? The position of a light switch.

What is the output of this C code?

```c
int main()
{
    int c = 2 ^ 3;
    printf("%d\n", c);
}
```

1

What is the output of this C code?

```c
int main()
{
    unsigned int a = 10;
    a = ~a;
    printf("%d\n", a);
}
```

-11

What is the output of this C code?

```
int main()
{
    if (7 & 8)
        printf("honesty");
    if ((~7 & 0x000f) == 8)
        printf("is the best policy");
}
```

is the best policy

What is the output of this C code?

```
void main()
{
    int x = 97;
    int y = sizeof(x++);
    printf("x is %d", x);
}
```

x is 97 (`sizeof()` 在编译时执行，`x++` 只是一个传递给 `sizeof()` 的表达式，它与传递 `x` 没有区别，不会实际执行 `x + 1`)

Week 03

Which of the following is a good reason (are good reasons) to equip the CPU with small amounts of fast memory?

- To make the design of the compiler simpler.
- To make some CPU instructions smaller
- To make some CPU instructions faster

II and III only

Which of the following must be true if a program is stopped at a specific line within the visual C++ debugger?

- There is at least one breakpoint enabled.
- There is a breakpoint enabled on that line.

- There is a breakpoint enabled on the line preceding that line.

none

Within visual C++, which of the following will reveal the value of a variable when the program is stopped at a breakpoint?

- Placing the mouse pointer over the variable name in the source file window.
- Inserting a printf() in the program.
- Typing the variable name on the "watch" window.

I and III only

Programs compiled for an Intel Pentium processor do not execute properly on a Sparc processor from sun microsystems because **the operation codes understood by the two processors are different**.

A branch instruction **sets the program counter to one of two possible values**.

A jump instruction **unconditionally sets the program counter to its operand**.

The machine code generated from source code by a compiler **does not preserve all the information given in the source code**.

Which of the following are true of the effect that optimizations have on the machine code generated by compilers?

- The resulting code will be faster and/or smaller.
- The resulting code will be clearer.
- The resulting code will be harder to debug.

I and III only

The program counter contains **the address of the CPU instruction that is about to be fetched**.

Immediately after the CPU executes an instruction that is neither a branch nor a jump instruction, the program counter **is incremented to point to the following instruction**.

Week 04

The visual C++ memory window displays **the contents of memory, interpreted in one of several ways, without the associated variable names**.

Consider the following code fragment.

```
int a;
int b;

int main(int argc, char *argv[])
{
    int c;
    int d;
    // some code
}
```

Which of the following must be true? **The value of `&d` is closer to the value of `&c` than to the value of `&a`.**

Consider the following code.

```
char a[100];
a[99] = *((char*) (((int) &a[0]) + 4))
```

If integers are 32 bits wide, which of the following values is equal to `a[99]`? `a[4]`

Which of the following statements about alignment within C struct's is true?

- Alignment may cause the allocation of unused space.
- Alignment is required by all modern processors.
- Alignment can help processors access data more efficiently.

I and III only

In C, assuming that an `int` takes 4 bytes, how many bytes are required to represent the array `int a[12]`? **48**

given the declaration and initialization `char s[] = "string"`, what is the value of the expression `s[6]`? `'\0'`

In C, assuming that an `int` takes 4 bytes, if array `a` is declared as `int a[12]` and a has the value `0x10000`, what is the value of the expression `a + 2`? `0x10008`

Given the address of a C struct at runtime, how is the address of a member element in the struct determined? **A constant offset associated with the member is added to the address.**

We want the variable `factorialfunc` to hold the address of the first instruction of the following function:

```
int factorial(int n)
{
    if (n == 1)
        return n;
    return n * factorial(n - 1);
}
```

How would we declare the variable? `int (*factorialfunc)(int)`

On one computer, the bytes with addresses `a`, `a+1`, `a+2` and `a+3` contain the integer `256`, and the variable declared with `int * a` has the value `a`. On a different computer, the bytes with addresses `b`, `b+1`, `b+2` and `b+3` also contain the integer `256`, and the variable declared with `int * b` has the value `b`. On a computer in which both addresses and integers are 32 bits wide, how many bytes of memory will the compiler allocate for following code fragment?

```
int a;
int* b = &a;
```

8

What is the output for the following code:

```
typedef struct
{
    char a;
    int b;
} test_struct_t;

int main(int argc, char *argv[])
{
    test_struct_t a, b;
    a.a = 0;
    a.b = 0;
    memset(&b, '\0', sizeof(b));
```

```
    if (0 == memcpy(&a, &b, sizeof(a)))
    {
        printf("struct a is equal to struct b");
    }
    else
    {
        printf("struct a is unequal to struct b");
    }
    return 0;
}
```

struct a is unequal to struct b

For the following piece of code:

```
typedef union
{
    long i;
    int k[5];
    char c;
} date;

struct data
{
    int cat;
    date cow;
    double dog;
} too;

int main(int argc, char *argv[])
{
    date max;
    printf("%d", sizeof(struct data) + sizeof(max));
    return 0;
}
```

What is the output for `printf()` statement? 52

Week 05

Consider the program given below.

```
int callee(void)
{
    int count = 5;
    printf("%d ", (int)&count);
    return count;
}

int main(int argc, char *argv[])
{
    int count = 4;
    count = callee();
    printf("%d ", (int)&count);
    return 0;
}
```

Two different integers are printed, and the value of neither can be determined from the information given.

What does the following program print?

```
int callee(int *count)
{
    count++;
    return *count;
}

int main(int argc, char *argv)
{
    int retval;
    int count = 4;
    retval = callee(&count);
    printf("%d", retval);
    return 0;
}
```

Cannot be determined from the information given.

What is printed as a result of execution of the following program?

```
void callee(int *count)
{
    (*count)++;
```

```
}

int main(int argc, char *argv)
{
    int count = 4;
    callee(&count);
    printf("%d", count);
    return 0;
}
```

5

What does the following program print?

```
void callee(int *count)
{
    (*count)++;
}

int main(int argc, char *argv[])
{
    int count = 4;
    callee(count);
    printf("%d", count);
    return 0;
}
```

Nothing: it will not compile successfully or popup access violation error message.

Consider the following program segment.

```
int factorial(int *arg)
{
    int n = *arg;
    if (n == 1)
        return n;
    return n * factorial(n - 1);
}
```

When the segment is executed, the variable `n` is allocated to **many addresses none of which is known to the compiler**.

At which of the following times is an activation record created?

- When a program starts executing.
- Every time a function is invoked.
- When a variable is declared.

I and II only

Consider the following program.

```
int i;
int j = 1;

int callee(int number)
{
    int plusone;
    plusone = number + 1;
    return plusone;
}

int main(int argc, char *argv[])
{
    if (j == 1)
        return callee(i);
    return j;
}
```

Which of the following are allocated in the activation record immediately after the function `callee()` is invoked? `plusone` and `number` only

Consider the following program.

```
int i;
int *jp = &i;

void main(int i, char *argv[])
{
    printf("%d %d\n", (int)&i, (int)jp);
}
```

Which of the following describes what it prints? Two very different integers.

Consider the following program.

```
int square(int *arg)
{
    int n = *arg;
    return n * n;
}

int main(int argc, char *argv)
{
    int arg = strtol(argv[1], null, 0);
    return square(arg);
}
```

When it is executed with the argument `5`, the variable `n` is allocated to **exactly one address not known to the compiler.**

Consider the following segment of C source code.

```
int a = 8;
int b = *&a;
```

What is the value of variable `b` at the end of execution of the segment? `a`

Week 07

Which of the following are true about statically allocated data in C programs?

- Its location is chosen by the compiler.
- Its location may change during execution if more memory is required.
- Its location is not known directly but can be found in a static symbol table.

I only

Consider a system in which memory consists of the following hole sizes in memory order:

```
H0  H1  H2   H3   H4  H5  H6   H7
10K 4KB 20KB 18KB 7KB 9KB 12KB 15KB
```

and a successive segment request of

- `12` KB

- `10` KB
- `9` KB

Which of the following sentences is true?

- *First Fit* algorithm allocates `H2`, `H0`, `H3` for the mentioned request.
- *Worst Fit* algorithm allocates `H2`, `H3`, `H7` for the mentioned request.
- *Best Fit* algorithm allocates `H6`, `H0`, `H5` for the mentioned request.

I, II, and III

Consider the `malloc()` function. Which one of the following sentences is correct?
The `malloc()` allocates the desired amount of memory on the heap.

In C, local variables allocated inside functions are allocated **on the stack**.

Suppose a compiler uses static storage to store all variables, function parameters, saved registers, and return addresses. Which of the following language features can this compiler support?

- Local variables.
- Function calls.
- Recursion.

I and II only

The key feature of implicit memory management is that memory is freed automatically. Which of the following features of C make(s) it difficult to add support for implicit memory management in C?

- Pointers are not always initialized.
- Type casting makes it impossible to know when a value could be a pointer.
- C programs can allocate memory at runtime.

I and II only

Which of the following features apply to standard heap allocation in C?

- The size of heap objects must be known at compile time.
- Heap memory must be explicitly allocated.
- Heap memory is deallocated when a function returns.

II only

In C, to allocate an array of 100 `long`s on the heap you should write `long* a = (long*) malloc(100 * sizeof(long))`

What is the value of an uninitialized pointer variable declared within a function? undefined

Consider the following fragment of C code.

```
int* p = (int*) calloc(100);
int* q = p;
free(p);
```

Immediately after executing it, which of the following are true about `p` and `q`?

- `p` and `q` are identical pointers to freed storage.
- `p` points to freed storage, and `q` points to an allocated block of size `100`.
- `p` should not be freed again, but invoking `free(q)` is all right.

I only

A memory leak is caused by a failure to free allocated memory.

In this sequence of C statements:

```
long a[10];
ptr = a + 5;
*ptr++ = x;
```

the last line could be rewritten as `a[5] = x; ptr = ptr + 1;`

Week 08

Explain the feature of stack. All operations are at one end.

When executing a function `callee()`, which of the following are true regarding the value of the frame pointer?

- It marks the top of the stack frame of the function that invoked `callee()`.
- It marks the bottom of the stack frame of `callee()`.
- It is the top of the stack.

I and II only

The C expression `a -> b` is equivalent to `(*a).b`.

What will be the output?

```
void main()
{
    char *p = "Hello world!";
    int *q;
    p++;
    q = (int *)p;
    q++;
    printf("%s%s\n", p, q);
}
```

`ello world! world!`

Why is it wrong to return the address of a local variable? **The variable address is invalid after the return.**

In C, when a struct is freed, **no pointers within the struct are freed automatically**.

To resolve memory leaks in C, one common approach is **to check whether the number of calls to `malloc()` is greater than the number of calls to `free()`**.

In C, `calloc()` differs from `malloc()` in that `calloc()` **sets the contents of the block to zero before returning**.

What properties of a variable are specified by the static keyword in C?

- The variable will be statically allocated.
- The variable name will be visible only to functions defined within the same file.
- The variable's value does not change very often. The compiler uses this fact to focus optimizations on other variables.

I and II only

A static variable by default gets initialized to `0`.

Week 09

When an array passed as an argument to a function, it is interpreted as **address of the array**.

What is the output of this C code:

```
void main()
{
    int x = 4;
    int *p = &x;
    int *k = p++;
    int r = p - k;
    printf("%d", r);
}
```

1

Which of the following operators below has the lower priority when evaluating? `*`

In one computer, the bytes with addresses `a`, `a+1`, `a+2` and `a+3` contain the integer `256`, and the variable declared with `int *a` has the value `a`. In a different computer, the bytes with addresses `b`, `b+1`, `b+2` and `b+3` also contain the integer `256`, and the variable declared with `int *b` has the value `b`. Which of the following are necessarily true?

- The contents of `a+1` are equal to the contents of `b+1`.
- The contents of `a+1` are equal to the contents of `b+2`.
- `*a == *b`.

III only

Consider the following function:

```
int factorial(int n)
{
    if (n == 1)
    {
        return n;
    }
    return n * factorial(n - 1);
}
```

How many activation records are "popped" when it is invoked by the expression `factorial(4)`? 4

A garbage collector **frees memory blocks that cannot be reached by dereferencing pointers**.

A memory pool is a large block of memory from which small objects are allocated piecemeal by breaking them off from the pool as required. Under which of the following conditions would such a scheme result in greatly improved performance?

- All objects allocated from the pool are freed at around the same time.
- All objects allocated from the pool are of similar sizes.
- A garbage collector takes care of freeing memory.

I only

Reference counts used in implementations of garbage collectors count **the number of pointers pointing to a block**.

How does Java handle memory allocation? **Java always uses a garbage collector.**

To quickly allocate and free many variables of a commonly used data type, we could **keep a linked list of free objects of that type's size**.

Mark-and-sweep garbage collectors are called conservative if **they treat everything that looks like a pointer as a pointer**.

Which statement is true? **Using the worst-fit algorithm on a free list that is ordered according to increasing block sizes is equivalent to using the best-fit algorithm.**

Which of the following statements are true?

- The worst-fit memory allocation algorithm is slower than the best-fit algorithm (on average).
- Deallocation using boundary tags is fast only when the list of free blocks is ordered according to increasing memory addresses.

none of them

The advantage of using copying GC including

- A copying collector is generally more efficient than a non-copying collector.
- The copying GC can make use of heap memory effectively.

I

A garbage collector starts from some "root" set of places that are always considered "reachable", such as

- CPU registers
- Stack
- Global variables

All of them

Week 11

In the process of software optimization process, what should do first? **find the hotspots**

Which of the following are useful for observing program performance?

- direct measurement with a stopwatch
- statistical sampling
- system monitors

I, II and III

Which of the following approaches towards optimizing programs is most advisable? **Optimize after all functions are written and debugged.**

What is TSC? **a timer mechanism of x86 platform, which is the shortname of time stamp counter**

Which of the following are advantages of using statistical sampling to profile programs?

- Exact run times of all functions can be determined.
- Code can be instrumented automatically.
- The performance impact due to measurement can be minimal.

II and III only

General wisdom, expressed by the 80/20 rule, says that **most execution time is spent in a small amount of code**.

Wall time measures **the total duration of a program's execution**.

CPU time measures **the time spent by a program executing program instructions**.

Amdahl's law, applied to program optimization, says that **successive program optimizations tend to produce diminishing returns**.

Which is a function call of C library? `clock()`

Week 12

Which of the following is likely to offer the best performance improvement for programs that spend 50% of their time comparing strings? **Store strings uniquely so that pointer comparison can be used.**

Which of the following is normal skill of making program run faster

- reducing procedure calls
- enhancing parallelism
- eliminating unneeded memory references

all

Which of the following is not optimization technique? **memory aliasing**

Which of the following is/are related to optimizing program performance by making it running fast

- by using faster algorithm
- by not using pointer
- by using data structure that ocupy less memory space

I only

Read the following code, and how can we optimize it?

```
void lower1(char *s)
{
    int i;
    for (i = 0; i < strlen(s); i++)
        if (s[i] ⩾ 'a' && s[i] ⩽ 'z')
            s[i] -= ('a' - 'a');
}
```

reducing procedure calls

On the following opinions of optimizing C programs, which is/are right?

- just config the compiler in its optimizing setting, then nothing else need to do
- understanding the feature of CPU is needless
- everything can be done in the C level, so it is needless to know the assembly code

none

Which of the following approaches towards optimizing programs is most advisable?
Optimize after all functions are written and debugged.

To quickly allocate and free many variables of a commonly used data type, we could **keep a linked list of free objects of that type's size**.

On profiling, which is/are WRONG?

- gprof is the profilling tool on linux platform
- it can be used to estimate where time is spent in the program
- it can incorporate instrumentation code to determine how much time the different parts of the program require

none

In order to optimizing program performance, we should know

- what is the hot spot
- understanding features of that processor on which the program will run
- all the system calls that the program uses

I and II only

Week 13

Compared to static ram (SRAM), dynamic ram (DRAM) is

- more expensive per megabyte
- slower per word access
- more persistent

II only

A memory hierarchy **takes advantage of the speed of SRAM and the capacity of disk**.

Which of the following is (are) true of the concept of locality of reference?

- It is used to predict future memory references precisely, with the help of the compiler.
- It is a quality of typical programs.
- It has been mathematically proven.

II only

Current technology trends suggest that the need for memory hierarchies **will never disappear**.

Which of the following levels of a typical memory hierarchy transfers data in chunks of smallest size? **CPU registers ↔ cache**

Which of the following levels of a typical memory hierarchy transfers data in chunks of biggest size? **main memory ↔ disk**

Which of the following manages the transfer of data between the CPU registers and the cache? **compiler**

Which of the following manages the transfer of data between the cache and main memory? **operating system**

Compared to dynamic ram (DRAM), disks are

- more expensive per megabyte
- slower per word access
- more persistent

II and III only

which of the following is necessarily true regarding the following code fragment?

```
a = b;
c = d;
if (e == 1) return;
```

It exhibits locality of reference no matter where the variables are allocated.

Week 14

Two computers A and B with a cache in the CPU chip differ only in that A has an L2 cache and B does not. Which of the following are possible?

- B executes a program more quickly than A.
- A executes a program more quickly than B.
- While executing a program, A fetches more data from main memory than does B.

I and II only

LRU is an effective cache replacement strategy primarily because programs exhibit locality of reference.

When the following code fragment is executed on a computer with 32-bit integers and a fully-associative cache with 32-byte cache lines, how many bytes of the array `a` will be fetched into the cache from main memory?

```
int a[100];
for (i = 0; i < 17; sum += a[i], i++);
```

at most 96

Your computer has 32-bit integers and a direct cache containing 128 32-byte cache lines. In the following code fragment, the compiler allocates `a` at address `0x800000` and `b` at address `0x801000`. Before the execution of the code fragment, the arrays `a` and `b` have never been used, so they are not in the cache. what is the minimum number of bytes from each of the arrays `a` and `b` that could be fetched into the cache from main memory, during the execution of the code?

```
int b[1024];
int a[1024];
for (i = 0; i < 17; sum += a[i] + b[i], i++);
```

1088

Which facts about the cache can be determined by calling the following function?

```
int data[1 << 20];

void callee(int x)
{
```

```
    int i, result;
    for (i = 0; i < (1 << 20); i += x)
    {
        result += data[i];
    }
}
```

- cache line size
- cache size
- cache speed

I only

Consider the following fragments from two versions of a program.

```
// version A
for (i = 0; i < n; i++)
{
    read(i);
    calculate(i);
    write(i);
}

// version B
for (i = 0; i < n; i++)
{
    read(i);
}
for (i = 0; i < n; i++)
{
    calculate(i);
}
for (i = 0; i < n; i++)
{
    write(i);
}
```

Which of the following are true of version B, compared to version A?

- B may be faster because of cache effects.
- B may be slower because of cache effects.
- B may execute at essentially the same speed as A.

I, II and III

About the cache in a computer system, which is true?

- Every computer system has 3 level cache, that is L1, L2, L3 cache
- Every computer systems' cache system have data cache and instruction cache
- Every computer systems' cache system has 2 level cache, that is L1, and L2 cache

none

A program whose code and data together occupy fewer than 256 KB is executed on a computer with a 512 KB direct cache. Which of the following is true? There is no telling, from the information given, how many bytes will be fetched from main memory.

Which of the following caches (all having the same size) will have the least number of conflict misses? a fully associative cache

When a cache is full and a new cache line needs to be fetched into it, which of the following is a pretty good, practical approach? Randomly selecting a cache location for the new line.

A certain program is found to execute with a cache hit ratio of 0.90 on computer A, and of 0.95 on computer B. However, because of other design parameters of these computers, its wall time is the same in both A and B. Then, a clever programmer finds a way to improve the locality of the program, so that it now executes with a hit ratio of 0.92 on A, and of 0.97 on B. Which of the following statements is valid? The wall time is now smaller on b than on a.

Week 15

From the time when a C program is written, to the time when it is running as a process on Windows, what should be done?

- compile
- link
- load

I, II and III

Read the following code in two C files.

```c
// a.c
extern int shared;

int main()
{
    int a = 100;
    swap(&a, &shared);
}

// b.c
int shared = 1;

void swap(int *a, int *b)
{
    *a ^= *b ^= *a ^= *b;
}
```

After compiling, about the relocatable object files, which is right?

- in the `.symtab` of `a.obj`, swap is `und`
- in the `.symtab` of `b.obj`, swap is `und`
- in the `.symtab` of `a.obj`, shared is `und`
- in the `.symtab` of `b.obj`, shared is `und`

I and III only

At what time can linking happen?

- compile time
- load time
- run time

I, II and III

What can linker do?

- resolution
- relocation
- take the same kind of sections from relocatable object files, and put them together according to their types

I, II and III

What can loader do?

- translate the C code into machine code
- resolution
- load or map the executable object file from the disk to memory

III only

Which variable will be put into bss?

```
int printf(const char format, ...);

int global_init_var = 84;
int global_uninit_var;

void func1(int i)
{
    printf("%d\n", i);
}

int main(void)
{
    static int static_var = 85;
    static int static_var2;
    int a = 1;
    int b;
    func1(static_var + static_var2 + a + b);
    return a;
}
```

- `a` and `b`
- `static_var`
- `global_init_var`
- `global_uninit_var`

IV only

Which file format is used for executable object file?

- pe

- coff
- elf
- a.out

all

Where the field, which describes whether relocatable object file is using little endian or big endian, locates? elf headers

Which section is used for resolution?

- elf header
- section header tables
- `.symtab`
- `.rel.text` and `.rel.data`

III only

Which section is used for relocation?

- elf header
- section header tables
- `.symtab`
- `.rel.text` and `.rel.data`

IV only

Week 16

What is right about trap?

- it is a kind of exception
- it can be used to implement system call
- it can be used to implement hard disk interrupt

I and II only

What is right about exception handler?

- it is used for handle exception
- it may not return
- it may return to the instruction where exception happen

I, II and III

What is right about exception?

- to handle it, hardware and software cooperation are needed
- it is just a hardware issue
- it is hardware platform dependent

I and III only

In ia32 or x86, the exception includes

- interrupt
- trap
- fault
- abort

all

In 32 or x86, which exception is used to implement system call

- interrupt
- trap
- fault
- abort

II

In 32 or x86, which exception is used to implement demand paging

- interrupt
- trap
- fault
- abort

III

About process, which one is right?

- by using process, we are presented the illusion that our program appears to have exclusive use of both the processor and the memory
- process is a running program

- process is possible by the help of exception

I, II and III

What is right about the process and thread of Windows

- process is the unit of resource ownership
- process is the unit of scheduling/execution
- thread is the unit of scheduling/execution

I and III only

Priority inversion is a situation in which **a high-priority thread indirectly waits on a lower priority thread.**

Which of the following levels of a typical memory hierarchy transfers data in chunks of smallest size? **CPU registers ↔ cache**