CSE101 Q&A

Author: ICS Strategy Group

Contents

Lecture 1

1. Mention four architectural(体系结构) levels of computer systems.

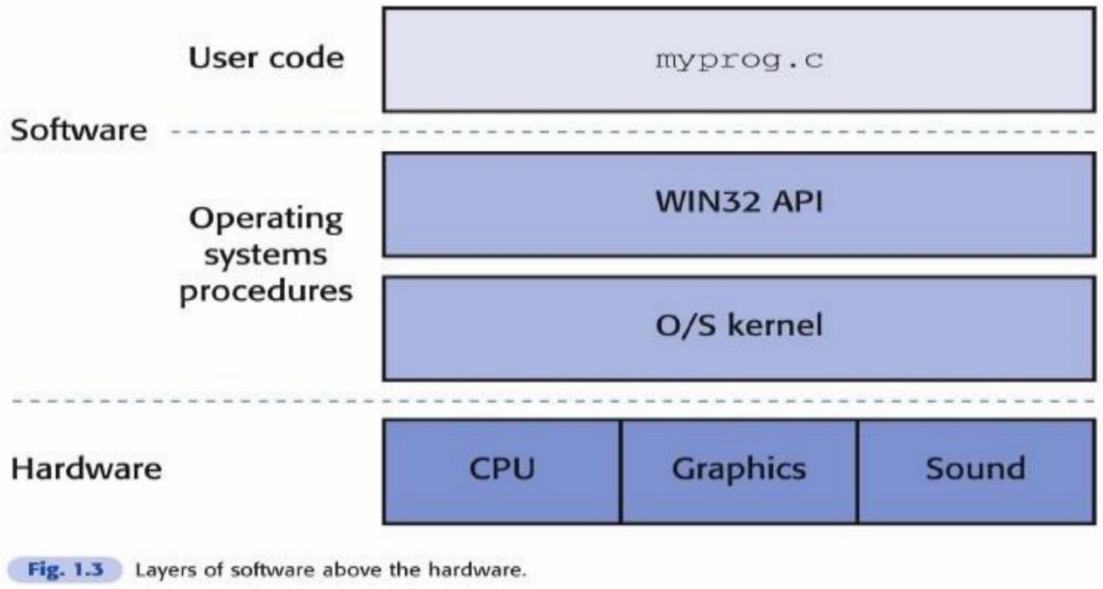


Fig. 1.3 Layers of software above the hardware.

2. What does CPU stand for? ALU?

CPU: Central Processing Unit 中央处理机

ALU: Arithmetic Logical Unit 运算器

3. What are the components of a computer system?

Input, output, memory, CPU.

4. Mention 4 different types of computers.

- Mainframe computers (1960s)
- Supercomputers (1970s)
- Workstations (1980s)
- Microcomputers (1980s)
- Personal computers (1980s)
- Microcontrollers (1980s)
- Servers (1980s)
- Chip computer

5. 'Servers' are facilitated via the availability of?

·network

·Web

·Cloud

6. Define the areas of study for 'computer systems'.

What is going on inside the computer when the programs are run.

How programs are written at low level (assembly programming)

7. Define 'Downward Compatibility'(向下兼容).

Most software written for computers with old hardware can be run on computers with newer hardware.

8. What is the description language that can be used to design high speed IC hardware?

VHDL （Very-High-Speed Integrated Circuit Hardware Description Language）

9. What are the advantages of having a hierarchical(分层) approach to computer system?

(1) For non-professionals, easy understand.
(2) For designer, easy to design.
(3) For hardware engineer(不确定), easy to change.

10. What is Moore's Law?

A circuit designed 24 months ago can now be shrunk to fit into an area of half the size.

11. **Functionalities of hardware systems can be brought out by what and thus offered to the user?**

Operating systems

12. What are the advantages of having 'operating systems' wrapping around the computer hardware?

– Ease of programming.
– Protection for the system and for other users.
– Fairness and efficiency of using system r

13. WIMP stand for? WIMP is available due to the development of?

– Window – Icon – Menu – Pointer

WIMP is available due to the development of hardware & software, mainly are input device, output device(like display).

14. What is the focus of scientific computing?

Computation

15. What is the focus of business computing?

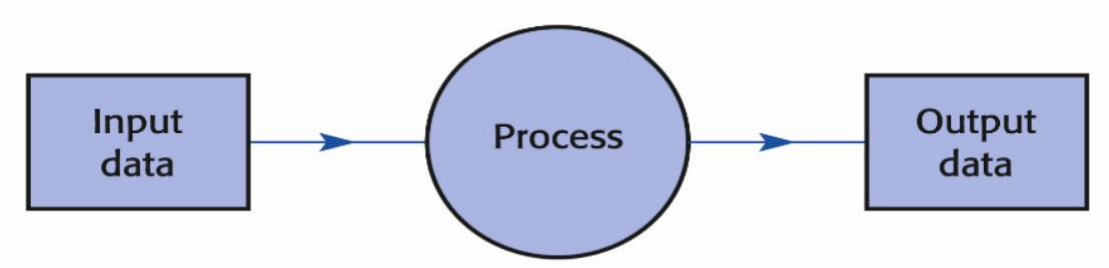
Data

16. What is the major characteristic of personal computing?

Interaction(交互作用)

Lecture 2

1. Using a diagram, illustrate the concept of Input-Process-Output Model.



2. Which model gives the fundamental structure of the current generation of digital computers?

The Input-Process-Output Model is the fundamental structure of the current generation of digital computers.

3. Process is controlled by what?

By program(software).

4. Highlight the three components required for the implementation(实施) of Input-Process-Output and von Neumann model.

– Hardware.
– Software.
– Data that is being manipulated

5. Name 5 examples of computer hardware.

CPU, memory, hard disk, keyboard, display screen

6. Identify the active part within a computer which performs calculations and other operations.

CPU

Central Processing Unit is an active part which performs calculations and other operations.

7. Which part of a computer holds data and programs for access by CPU?

The main memory or system memory holds data and programs for access by CPU.

8. Give 3 examples of secondary storage.

Hard disk, CDs, DVD

9. Give 3 examples of input devices.

Keyboard, Mouse, Scanner（扫描器）

10. Give 3 examples of output devices.

Monitor（显示器）, Speaker（扬声器）, Printer（打印机）

11. Define 'software'.

Software is a collection of program, and can be design to carry out to execute some specific processes.

12. Difference between these two models Input-Process-Output model & von Neumann model?

Von Neumann is the 'process' part of the input-processs-output model.

13. For a particular machine, the machine instruction set is usually fixed. True or false?

True.

For a particular machine, this set is fixed for specific purpose.

Every CPU has its own instruction set (100-200 instructions, typically).

14. There exists a standard instruction set for industry purpose. True or false?

False

Although the instruction sets of different CPUs are similar, there is no standard instruction set.

15. High Level Programming Languages (HLLs) are more suitable for programming than the languages of machine instructions. Why?

(1) More user friendly.

(2) More convenient for programming.

(But the programs in HLLs still have to be translated to the machine codes)

16. Mention 4 major categories of machine instructions.

Data transfer and manipulations

Input-output

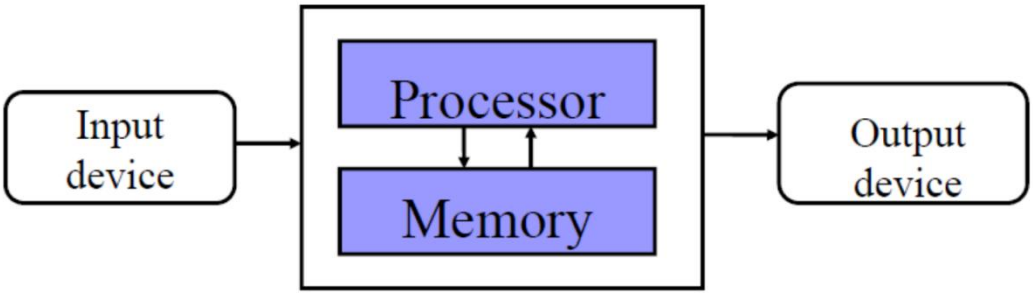
Program control or hardware control

Machine control

17. **Why HLL programs need to be translated before execution?**

Because CPU can only understand machines codes

18. What is the von Neumann model? (Very Important)



19. Both ? and ? are held in computer's memory (store) and both represented by?

Both program and data are held in computer's memory (store) and both represented by binary codes.

20. Identify the von Neumann bottleneck.

Where: Interconnection between the CPU and main memory.

Cause: CPU fetch speed faster than main memory.

21. How could computers distinguish data from instructions since they are both represented by binary codes?

Because computer know where to fetch data and where to fetch instructions.

22. What is the main difference between von Neumann machine and Harvard architecture?

Harvard architecture separates(分离) data from programs, and in von Neumann machince, data and instructions shared the same memory.

23. Motivate the use of Harvard architecture.

Increase transfer rates, improving throughput.

24. What is the additional cost from the usage of Harvard architecture?

Data and instruction need single reservoir(储存器).

25. Most desktop CPUs have an internal(内部的) 'instruction cache(指令缓存)' feeding the control unit and a completely separate(独立) 'data cache'. This mimicks(模仿) which computer architecture?

Harvard architecture.

Lecture 3

1. Name 4 examples of HLLs.

Java, C, Python, C++

2. Translation fills in the semantic gap in computer systems. (True or false?)

True

3. Name 3 different ways of translation.

Interpreting(解释), Compiling(编译), Assembli(汇编)

4. Identify the crucial role(关键作用) of translation under each.

Compilers, translating HLL instructions into machine code before the code can be run on the machine.

Assemblers, translating mnemonic form of machine instructions into their binary codes.

Interpreters, translating HLL instructions into machine code on-the-fly (while the program is running).

5. When compile-time errors occur, what do we do?

Go back find the source program and debugged.

6. **What are the purposes of linking?**

–The linker is to join together all the binary parts.

–The linker will report errors. (When it cannot find the module or code referred to by those external references.)

7. 'Loading' is carried out before 'linking' after a program is compiled. (True or false?)

False.

Loading is after linking.

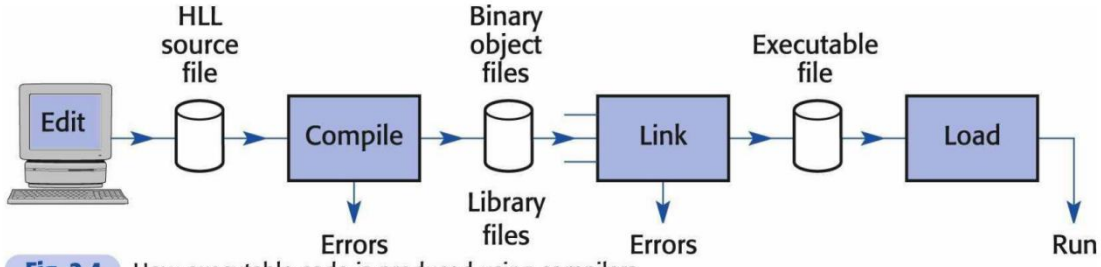


Fig. 2.4 How executable code is produced using compilers.

8. Program modules can be compiled separately(单独). (True or false?)

True. Each module has to be designed, coded and compiled.

9. A compiler can translate a module into binary codes, but it cannot resolve those references to other modules. This occurs when …?

When a program is separate into different modules.

10. Library files are usable if linked into your program code. (True or false?)

True. If it not link you cannot link.

11. Interpreters typically convert program code into what?

Intermediate code.(中间代码)

12. What is the output of program compilation?

Binary machine code.

13. Name 2 scenarios(场景) where interpreters(解释器) are more useful than compilers(编译器).

(1) When frequently change the program.
(2) Want program run fast.
(3) The program needs to be optimized in real time (Interpreters are more accurate in terms of error reporting).
(4) Easily portable, need programs that can be easily ported to other platforms (Interpretation can provide uniform execution environment across several diverse compute).

14. Interpreters are sometimes called as virtual machine because ...?

Because it decoded and executed one instruction at one time like machine cycle.

15. Mention 3 approaches to code sharing.

Source-level subroutines and macro libraries(宏程序库).
Pre-translated re-locatable binary libraries.
Dynamic libraries and dynamic linking.

16. Name 2 disadvantages (or issues) or macro libraries.

Who owned the code?
Who should maintain it?

17. Libraries can be linked into your program code, but not altered. (True or false?)

True.

18. What are the disadvantages of program execution with pretranslated program library?

Each program is to have a private copy of the subroutines(子例程), wasting valuable memory space, and swapping time, in a multitasking system.

19. There exists a de-facto standard(事实标准) of dynamic libraries and dynamic linking. (True or false?)

True.
(What fact? It's Microsoft active as standard.)

Lecture 4

1. **Define 'bit'.**

   A bit is the most basic unit of information

   It contains the information necessary to distinguish two alternatives (1 or 0, YES or NOT, etc.).

   1B = 8bit

2. **Decimal notation(十进制) is more compact(紧凑的) than binary. (T or F).**

   True.

3. **Binary is more "convenient" for computers due to?**

   Based on the two-state, ON/OFF technology

4. **Every number can be used as a base. (T or F).**

   True. (Except 0 and 1.)

5. **Hexadecimal notation uses ? as a base.**

   Hexadecimal notation uses 16 as a base

6. **Octal notation uses ? as a base.**

   Octal notation uses 8 as a base.

7. **Why do we use hexadecimal, or base 16, number notation?**

   Convenient shorthand for binary numbers.

   Every hexadecimal digit exactly represents 4 binary bits

8. **How many bits are required to encode ASCII?**

   7 bit

Lecture 5

1. The most widely used binary code with non–IBM mainframes and virtually all microcomputers, is

a. EBCDIC b. DOS c. ASCII d. LAN

c. ASCII (American Standard Code for Information Interchange)

2. **Which of the following coding schemes is not yet commonly used?**

a. Unicode b. EBCDIC c. ASCII d. All of the above are common coding schemes.

b. EBCDIC (Extended Binary Coded Decimal Interchange Code)

EBCDIC appeared first, and was replaced by ASCII later. Encoding and sorting are different, and the two can be converted to each other before

3. **A drawback to ASCII is that it**

a. cannot handle all the characters of some languages other than English and European languages.

b. uses only 4 bits to form each character.

c. is slower than EBCDIC.

d. None of the above is correct

A True

B Wrong. ASCII (7-bit code) and its extensions (8-bit codes)

C Wrong. There is no difference between fast and slow encoding. The encoding sequence of EBCDIC is discontinuous, which is more difficult to remember than ASCII.

4. ASCII divided into two classes of codes?

Printing characters & Control characters

5. What isthe most widely-used standard for floating-point computation?

IEEE 754 Standard

6. Two things happen when you declare variables in a program. What are they?

(1) You are telling the compiler to reserve the correct amount memory space to hold the variable.

(2) You are also telling the compiler what encoding/decoding/representation scheme to be used.

Lecture 6

1. What is the most important development during the past 40 years evolution of OS?

Window interface or WIMP.

2. OS needs to provide fair & protected access to system resources, why?

(1) Resources are improperly managed and multiple processes can deadlock
(2) At the same time, it can protect the hardware from improper operation by users

3. Explain 'multi-tasking'.

Program can run simultaneously, if they are not too resource consuming

4. Mention 3 functions that need to be provided when an OS is to support 'multitasking'.

(1) Memory management
(2) Security(安全) (A program cannot change other program's code or data)
(3) Allocate CPU time to each program

5. Explain 'multi-user'

It is a system can support multiple users run program at same time.

6. How does operating system provides access to network facilities?

Via networking API (Application Program Interface 应用程序界面)
For example: socket interface

7. Telecommuting or online shopping are enabled through the introduction of?

Phone, Fax, and/or Computer

Lecture 7

1. What is the difference between a 'general purpose machine' and a 'special purpose machine' ?

General purpose machine have Operation System. Can do anything but the efficiency is not very high.

Special purpose machine don't have Operation System. Can do and only can do some special things efficiently.

2. **Assume there are 6 devices to be interconnected via 8 data lines (wires) plus 2 control lines (wires), how many wires will be needed if point-to-point connection scheme is used?**

(5*6)/2 * (8+2) = 150 (Each point to point need 8+2 lines)

3. Name 2 registers that are always used during each instruction execution.

(1) IP(Instruction Pointer)
(2) IR(Instruction Register) or PC(Program Counter)

4. **How many pixels are rendered with 1024 x 768 screen resolution?**

786432

5. Name 2 typical applications run by coprocessors.

math
graphics

6. Name 3 different types of bus in a computer system.

address bus
data bus
control bus

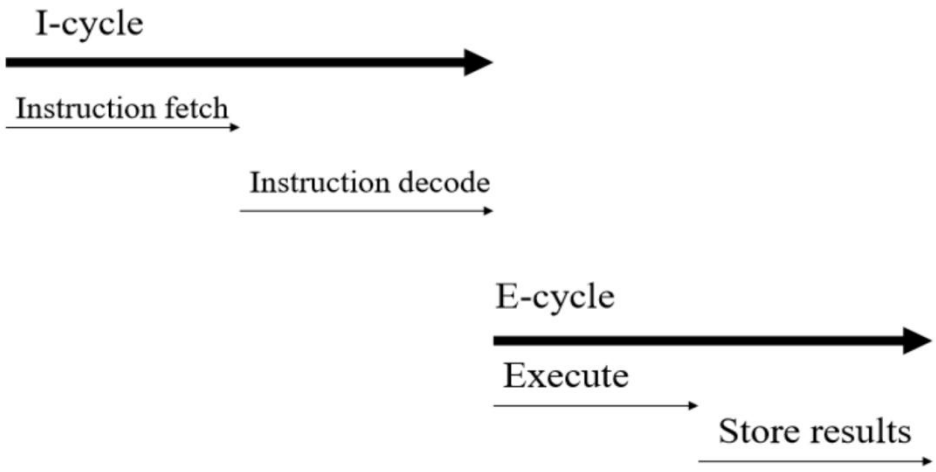
7. What is the reason behind bus bottleneck?

(1) Speed did not match between CPU and main memory
(2) A bus can only transfer one item of data at a time

8. **Highlight the 2 major components of a CPU.**

ALU (Arithmetic and Logic Unit).
Control Unit

9. **What tasks are performed during a machine cycle.**



10. **Computers & printers are usually connected via what type of ports?**

Past: Parallel port

Now: USB port

11. RISC refers to

a. RAM that supports fewer instructions than do CISC chips.

b. instructions that support fewer codes than do CISC chips.

c. processors that support fewer instructions than do CISC chips.

d. coding schemes that are used as a back–up to CISC.

C

CISC

–complex instruction set

–most mainframes and PCs

RISC

–reduced instruction set

–cheaper and faster

–shift some work to software

RISC uses simple word, and CISC often uses more words to express the same instructions

12. The computer's main processor follows ? instructions to manipulate data into information.

a. hardware b. CPU c. software d. Unicode

c. software

13. This type of hardware consists of devices that translate data into a form the computer can process.

a. application b. input

c. system d. None of the above is correct.

b. input

14. Name 4 registers that are always used during each instruction execution

IP(Instruction Point), IR(Instruction Pointer), MAR(Memory Address Register), MBR(Memory Buffer Register)

Lecture 8

1. The execute phase is the same for different types of instructions. (T or F?)

False.

The fetch phase is the same for different types of instructions/

The execute phase depends on the type of instruction.

2. The assembler program takes as input a program written in assembly language, and translate instructions into ? form.

Translate instructions into their binary machine-code form.

3. Assembler associates labels with ?

Assembler associates labels with memory addresses.

4. A CPU may execute only instructions loaded in the main memory (T or F ?).

True.

5. Access time is the same for all the RAM cells (T or F ?).

True.

Access time is the same for all the stored items（RAM cells）

6. What is the meaning of MOV EBX, [EBX] ? What is the side effect of this instruction?

Assign the contents of the address pointed to by the pointer in the current EBX to EBX.(把当前 EBX 里的指针指向的地址所包含的内容赋值给 EBX。)

More Information:

https://zhidao.baidu.com/question/161765829.html

7. JNZ L2 means ? When will there be compile time error for the above instruction?

Jump to label L2 when the value is not zero.

When the label L2 is missing.

8. Is this valid assembly code? MOV maxval, loc1

Illegal.

It's illegal from memory to memory.

Lecture 9

1. Name 3 use cases of the Flag register in Pentium.

Zero flag – set when destination equals zero
Sign flag – set when destination is negative
Carry flag – set when unsigned value is out of range
Overflow flag – set when signed value is out of range

2. The Flag register can be used to pass information between one instruction and the subsequent instructions. (T or F?)

True

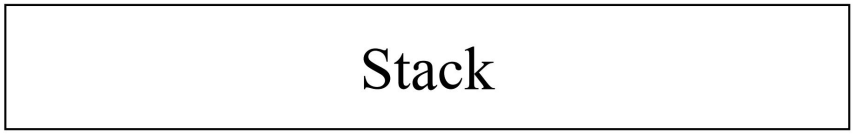
3. Which register is used to store the result of subtraction from this instruction? CMP AL, BL

The result of subtraction in CMP instructions will not be saved by registers, but the result will affect the flag register EFlag.

The essence of CMP AL, BL is AL - BL, and affects the value of flag register EFlag through the result.

4. **Under inline assembly, what mechanism(机制) is used to pass arguments(参数) to printf routine(例程) ?**

Stack



- **Stack** is a memory arrangement (data structure) for storing and retrieval information (values).
- The order of storing and retrieving the values for the stack can be described as **LIFO** (last in, first out).
- The **ESP** register stores the address of the item that is on top of the stack.
- Example of an alternative memory arrangement is a **queue**: **FIFO**.

5. Inline assembler can be used to call a C library function within an assembly segment in a C program (True or False).

True

6. Status flags are set (or cleared) before an instruction is being executed. (True or False)

False.

It's after, not before.

7. D Flag is used to set the direction of looping.

False.

EFLAG: The Flag register holds the CPU **status flags**.

The status flags are separate bits in EFLAG where information on important arising conditions such as **overflow** or **carry bits**, is recorded.

ID	Identification flag or CPUID availability
VIP	Virtual interrupt pending
VI	Virtual interrupt active
AC	Alignment check
VM	Virtual 8086 mode active
RFR	Resume task after breakpoint interrupt
NT	Nested task
IOPL	IO privilege level
O	Arithmetic overflow error
D	Direction of accessing string arrays
IE	External interrupt enable
T	Trap, single-step debugging, generates an INT #1 after each instruction
S	Sign, MS bit value
Z	Zero, result being zero
A	Auxiliary carry, used by BCD arithmetic on 4 LS bits
P	Parity, operand status
C	Carry, indicates an arithmetic carry or borrow result

Fig. 7.9 CPU status flags.

CPU status flags

- Most often used flags are:
  - S: sign.
  - Z: zero, result being zero.
  - C: carry, indicates an arithmetic carry.
  - O: arithmetic overflow error.
    - The addition of two large positive numbers which may give a negative result in a Two's complement system.

Lecture 10

1. The Pentium instruction set has a fixed width for all instruction to speed up instruction decoding.

False.

Pentium instructions can be from 1 to 15 bytes long, depending on the operation required and the address modes employed.

2. Which are those 3 major components covered by every instruction?

– The operation required.

– The location of operands(操作数) and results.

– The data type of the operand

3. **Machine instructions are encoded with what to contain information about the operation required?**

Distinct bit fields

4. Which of the following is the fastest to execute?

A. mov eax, b B. mov a, ebx C. mov eax, ebx

C

Data Register Direct is the fastest to execute.

5. The following instruction will involve the calculation of 'effective address'.

mov eax,table[esi]

True.

The effective address is obtained by adding the address 'table' contained in the operand field of the instruction to the content of a register (registers).

When read the value in table[esi], first get the beginning address of the table and then add esi to get the needed address. Then get the value in the address.

6. After the execution of the following instruction, eax will contain the content of message1 inside.

lea eax, message1

False

EAX will contain address ,not the value.

Lecture 11

1. To pass two parameters to printf in inline assembly code, the first parameter should be pushed onto the stack first. (True or false)

False

The second parameter should be pushed onto the stack first.

According to FILO(First In Last Out) principle.

2. When calling printf in inline assembly, the parameters passed to it will be popped off stack by printf. (True or false)

False.

Printf won't pop, should add 'add esp' by users.

3. When calling scanf in inline assembly, the address of the variable to receive input needs to be pushed onto the stack. (True or false)

True.

4. What is the conversion(转换) specifier(说明符) to be used when printing a string under printf?

%s

– %c print a character

– %d, or %i print a signed decimal number

– %s print a string of characters

5. If three integer parameters were pushed onto stack when calling 'scanf' in inline assembly, how would you adjust the value of register 'esp' when returning from 'scanf'?

add esp, 12

(push n times, add esp 4*n)

6. **The execution of cmp eax,ebx will check upon the setting of zero flag. (T or F)**

False, cmp will change the ZF(Zero Flag), not check.

Lecture 12

1. Conditional(条件) jumps in assembly can be used to implement(实现) HLL constructs like while, for and switch. (T or F)

True.

2. 'loop' instruction in assembly has a branching effect based upon the value of decremented(递减的) ECX register. (T or F)

True.

In the loop, ECX automatically subtracts 1 after each loop.

3. Explain what 'LOOPNE label' does.

Loop the program in label if not equal.

4. Explain what the following instructions do.

CMP EAX, EBX

LOOPNE label

If EAX and EBX is not equal, then loop the program in label.

Important Exercise:

Implementing higher-order constructs: conditional statements

- In Java:

```
if (c > 0)
  pos = pos + c;
else
  neg = neg + c;
```

- Equivalent in the assembly code:

```
     mov eax,c
     cmp eax,0
     jg positive
negative:add neg,eax
     jmp endif
positive:add pos,eax
endif:…
```

Implementing higher-order constructs: conditional statements

- In Java:

```
if (c > 0)
  pos = pos + c;
else
  neg = neg + c;
```

- Equivalent in the assembly code:

```
     mov eax,c
     cmp eax,0
     jle negative
positive:add pos,eax
     jmp endif
negative:add neg,eax
endif:…
```

Switch-Case statement

- In Java:

```
switch (num){
  case 1:  … ;
     break;
  case 2:  … ;
     break;
}
```

- Equivalent in the assembly code:

```
mov eax,num
     cmp eax,1
     je case_1
     cmp eax,2
     je case_2
     jmp end_case
case_1:…
     jmp end_case
case_2:…
end_case:…
```

Implementing higher-order constructs: the *for* statement

- In Java:

```
for (int x = 0;
  x < 10; x++)
{
  y = y +x;
}
```

- Equivalent in the assembly code:

```
     mov eax,0
for_loop:add y,eax
     inc eax
     cmp eax,10
     jl for_loop


Problem of this
  implementation?
```

Implementing higher-order constructs: the *while* statement

- In Java:

```
while (fib2 < 1000)
{
  fib0 = fib1;
  fib1 = fib2;
  fib2 = fib1 + fib0;
}
```

- Equivalent in the assembly code:

```
while:mov eax,fib2
     cmp eax,1000
     jge end_while
     mov eax,fib1
     mov fib0,eax
     mov eax,fib2
     mov fib1,eax
     add eax, fib0
     mov fib2,eax
     jmp while
end_while: …
```

Implementing 'loop'

- Loop in an assembly code:

```
     mov ecx,200
next:…
     …
     …
     loop next
```

- Can we do without 'loop'?
- Hint: use 'cmp', 'dec', 'jne'
- Equivalent without loop construction:

```
   mov ecx,200
next:…
   …
   …
   dec ecx
   cmp ecx,0
   jne next
```

Implementing *for* statement : Exercise		*do-while* statement : Exercise	

• In Java:
`for (int x = 3;`
`  x < 20; x=x+2)`
`{`
`  y = y +x;`
`}`
		• In Java:
		`do{`
		`  fib0 = fib1;`
		`  fib1 = fib2;`
		`  fib2 = fib1 + fib0;`
		`}while (fib2 < 1000)`
			• Equivalent in the assembly code:

The exercise in for statement:

```
    mov     eax, 3
for_loop:
    add         y, eax
    inc     eax
    inc     eax
    cmp     eax, 20
    jl      for_loop
```

The exercise in do-while statement:

```
while:
    mov     eax, fib1
    mov     fib0, eax
    mov     eax, fib2
    mov     fib1, eax
    add         eax, fib0
    mov     fib2, eax
    mov     eax, fib2
    cmp     eax, 1000
    jge     end_while
    jmp         while
```

Lecture 13

1. A subroutine(子程序) can be called at various moments from different places in the main program. (T or F)

True.

2. A subroutine can be called by itself. (T or F)

True.

3. **How many return addresses can you store using a stack?**

As many as you want. Because the stack is efficiency, you cannot running out all the stack space.

4. How many nested calls can you make within a program?

As many as you wish.

5. How does the computer know when to return from a subroutine?

Through the return instruction 'ret'.

6. What happens when a 'CALL …' instruction is executed?

(1) Copy the address from EIP (Instruction Pointer) to the stack (PUSH).

(2) Puts the required subroutine address into EIP (Instruction Pointer)

7. What happens when a 'RET' instruction is executed?

Pop the last address stored in the stack and put it into EIP.

Lecture 14

1. **When is value parameter(参数) good for?**

   When you only need value.

Value parameters

- In many cases, the additional information required by a subroutine will be a simple **value** (or values) of some type(s):
  - For example, a numeric value, or ASCII code value, etc.
- Imagine a subroutine, when given two numbers, has to decide and return which number is bigger. All this subroutine needs is the values of two variables.
- Such parameters are called **value parameters**.

2. **When is reference parameter good for?**

   When you need the address of the parameter of the each address of the memory value.

3. **Why multiple stack frames(帧) can coexist(共存) at the same time?**

   Because of the nested calls several stack frames may be present at the same time.

4. **What type of data are stored under a stack frame?**

   (1) Parameters of the subroutine.
   (2) Return addresses.
   (3) EBP(Extended Base Pointer / Stack Frame Base Pointer) for your all stacks frame
   (4) Local variables

Important Exercise:

An example: value parameters

```
        …
        mov eax, first          ;
        mov ebx, second         ;
        call bigger             ; calling
        mov max, eax            ;
        …
bigger  proc                    ; procedure which uses value parameters
        mov save1, eax          ; passed through registers EAX and EBX
        mov save2, ebx          ;
        sub eax, ebx            ; the result of the procedure is returned
                                ; again via register EAX
        jg first_big            ;
        mov eax, save2          ;
        ret                     ;
first_big: mov eax,save1        ;
        ret                     ;
bigger  endp                    ;
```

Exercise

- Write a subroutine: when given two numbers in EAX and EBX, it returns their difference via EAX.

```
    ...
    mov     eax, num1
    mov     ebx, num2
    call    difference
    mov     diff, eax
    ...
difference proc
    cmp     eax, ebx
    jg      first_big
    sub         ebx, eax
    mov     eax, ebx
    ret
first_big:
    sub         eax, ebx
    ret
difference endp
```

Exercise

- Write a subroutine: when given two variables, it swaps their values.

```
    ...
    mov     eax, num1
    mov     ebx, num2
    call    swap
finish:
    ...
swap proc
    mov     temp, eax
    mov     eax, ebx
    mov     ebx, temp
    ret
swap endp
```

An example: reference parameters

```
      …
      lea eax, first          ;
      lea ebx, second         ;
      call swap               ; calling
finish: …
  ;
swap proc                     ; subroutine which uses
      mov temp, [eax]         ; reference parameters
      mov [eax],[ebx]         ; passed via registers
      mov [ebx],temp          ;
      ret                     ;
swap endp
```

Lecture 15

1. A recursive(递归) procedure(过程) will typically provide an exiting condition. (T or F)

True.

2. A recursive procedure will typically have a divide-andconquer step. (T or F)

True.

3. **What is the side-effect of the procedure 'multiply'?**

(1) Stack modified (2 pops)

(2) eax original contents replaced with the product

4. A recursive procedure can always be re-implemented using iteration(迭代) without recursion. (T or F)

True.

Important Information:

https://blog.csdn.net/idaaa/article/details/78155553

Lecture 16

1. What are the pros & cons of BCD?

Cons:

BCD is less economical than binary representation

Calculations in BCD are more difficult

Pros:

Translation between BCD and character form is easier

Unsigned integers: BCD	Binary vs. BCD representation
• **Binary-coded decimal** (BCD): – Digit-by-digit binary representation of original decimal integer. • Each decimal digit is **individually** converted to binary. – This requires **4 bits per digit** (not all 4-bits patterns are used). • Example. – Decimal value 68 is presented in BCD as 01101000. • One byte can store **unsigned integers** in a range 0-99 under BCD encoding (Why?).	• BCD is less economical than binary representation (Why?). • Calculations in BCD are more difficult. • But, translation between BCD and character form is easier. – Last 4 bits of ASCII numeric character codes correspond precisely to the BCD representation. • Example. – ASCII code of '5' is 0110101, its BCD representation is 0101. • Binary representation is more common, BCD is used for some business applications.

2. Is it possible to have a 3's complement scheme?

Yes.

3. **Can overflow still occur under 2's complement addition?**

Yes.

For example, when two positive numbers add up, the result is negative, the overflow happened.

4. What are the pros & cons of 2's complement?

Cons:

From a human point of view, they are not easy to understand.

Pros:

They are easy to implement(实现) bases on two-state technology.

5. Try to implement 10's complement in 2 decimal digits

(14 marks) Complete the following binary addition. What is the decimal equivalent of this addition? Identify problems in this computation. Assume the operands are prescribed in 2's complement format and only 7 bits are available. Suggest a solution in details for such a scenario.

```
  1001101
+ 1010111
  ???????
```

```
  1001101
+  1010111
 01000100
```

This binary addition involves the addition of two operands: 1001101 (-51) and 1010111 (-41). The result (01000100) from this binary addition gives 36 in decimal, which is not what we expect from (-51)+(-41) = (-92). The problem arises because of this addition

runs out of 7-bit capacity for the sum. (8 marks)
Solution: Use an 'overflow' flag mechanism to detect such an overflow, i.e. use the following simple rule to detect overflow: If both inputs to an addition have the same sign, and the output sign is different, overflow has occurred. (6 marks)

Lecture 17

1. Addition for n-digit numbers represented by 10's complementary convention(complementary convention,附约) is done based upon addition modulo n. (T or F)

False. It should modulo 10.

2. **How is overflow detected during addition?**

If both inputs to an addition have the same sign, and the output sign is different, overflow has occurred

3. **Can overflow result from subtractions**?

Yes, the subtractions are another way to express addition, addition overflow occur, subtractions overflow occur.

4. Under 2's complement system, numbers that begin with 1 are representing negative numbers. (T or F)

True.

5. How many bits are required for a Java short integer?

16 bits.

6. **How many bits is required for a Java long integer?**

64 bits.

7. An 'int' type under Java encodes integers under this range: -2^{32} --- $2^{32}-1$ (T or F)

False.

- **byte** 8-bit: integers from -2^{8-1} to $2^{8-1}-1$.
- **short** 16-bit: integers from -2^{16-1} to $2^{16-1}-1$.
- **int** 32-bit: integers from -2^{32-1} to $2^{32-1}-1$.
- **long** 64-bit: integers from -2^{64-1} to $2^{64-1}-1$.

Lecture 18

1. Under the IEEE 754 standard..

– how many bits are required to specify the sign?

– how many bits are required to specify the sign of the magnitude?

– how many bits are required to specify the sign of the exponent?

1 bit for sign.

2 bits for magnitude

4 bits for exponent

Floating point formats

- Any format for floating point numbers should specify how the components of an exponential notation are stored (in a word, or several words).
- The **base of the exponent** and the **location of the binary point** are standardised as part of the format and, therefore, **do not have to be stored** at all.

Floating Point Formats (cont.)

- **Example:** Suppose, that the standard code consists of space for seven digits and a sign:

  SEEMMMMM

  – So, we have two digits for the **exponent** and 5 digits for the **mantissa**.
- Trade-off: precision (mantissa) vs. range (exponent).
- Most commonly, the mantissa is stored using **sign-magnitude** format.
- What about the sign of the exponent?

2. **Under the IEEE 754 standard, how many bits are required to specify the decimal point position?**

No need.

3. Does IEEE standard 754 provide a specification for NaN?

Yes. When exponent equal 128.

(NaN means Not a Number)

Exponent biasing

- The exponent is biased by 2^{8-1}-1, that is, biased by 127.
- Exponents in the range -127 to +127 are representable.
- e=128 reserved for NaN, infinity

4. Under the IEEE 754 standard for single precision(精度) floating point format, what type of excess notation(过剩表示法) is used for exponent specification?

127.

5. Under the IEEE 754 standard for double-precision floating point format, what type of excess notation is used for exponent specification?

1023.

Lecture 19

1. A computer system encodes data by means of this coding scheme to represent letters, numbers, and special characters.

a. magnetic (磁性的)

b. analog （模拟）

c. binary （二进制）

d. optical （光学的）

C

2. Which of the following is volatile?

A. ROM

B. RAM

C. DVD

D. Hard Disk

B

Both types of RAM are volatile, because they lose their contents when the power is turned off.(Lecture19/P13)

3. Which of the following has best access?

A. register

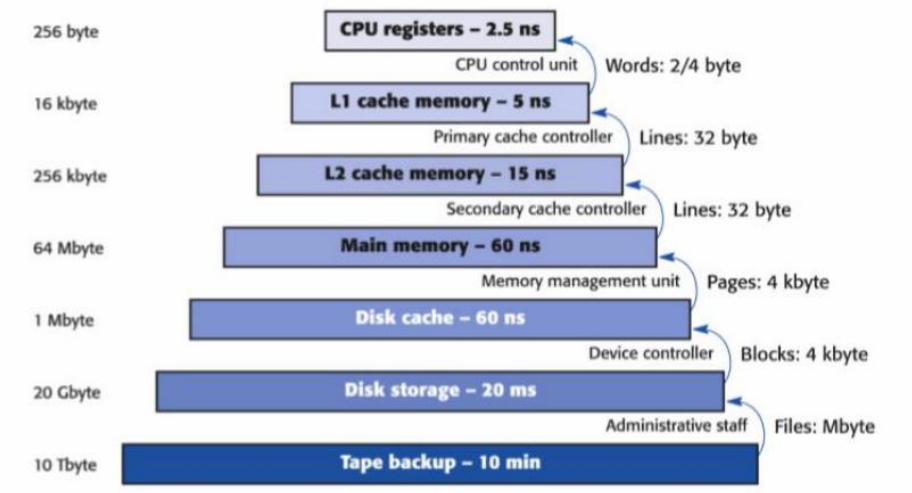
B. cache

C. DVD

D. RAM

A

Register is in CPU, which has very fast access.



Going down the hierarchy, the access time is increasing, but the capacity is increasing.

(cache, 缓存)

Lecture 20

1. Which of the following has fastest access?

A. register

B. cache

C. disk

D. tape

A

2. **Which of the following is most expensive in terms of cost?**

A. register

B. cache

C. disk

D. Tape

A

3. **Given the following components, form a reasonable memory hierarchy to ensure good cost-speed tradeoff(权衡).**

Cache, disk, RAM

Ans:

Cache, RAM, disk

4. Locality(位置) can be temporal(时间), which means data that are recently used tend to be accessed more likely than the others. (T or F)

True.

Lecture 21

1. When you save to this, your data will remain intact even when the computer is turned off.

a. RAM

b. motherboard

c. secondary storage device

d. primary storage device

C

A secondary storage device is a non-volatile device that holds data until it is deleted or overwritten.

A primary storage device is a medium that holds memory for short periods of time while a computer is running.

RAM is a volatile memory and requires power to keep the data accessible. If the computer is turned off, all data contained in RAM is lost.

2. This data access method will slow down the process of data retrieval.

a. direct access storage

b. sequential(顺序) storage

c. random access storage

B

3. The closed, concentric rings(封闭的同心圆) on a diskette(磁盘) are referred to as

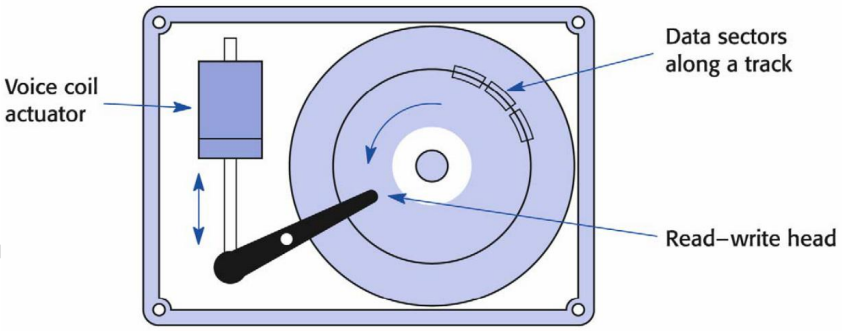
a. grooves.

b. tracks.

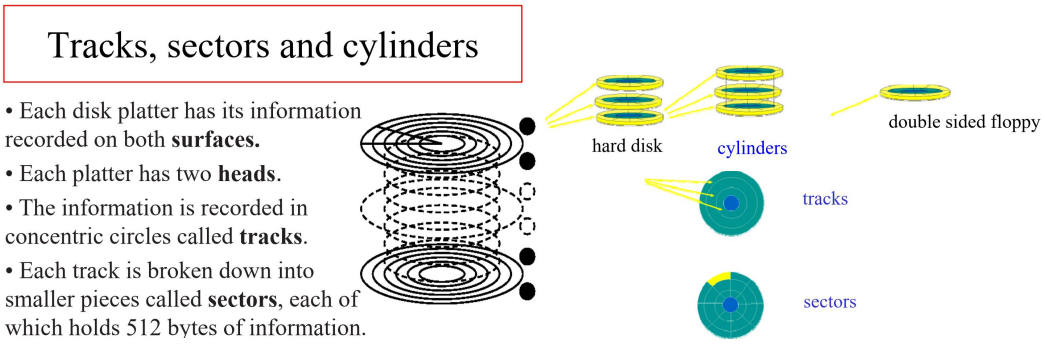
c. sectors.

d. circles.

B



(voice coil actuator, 音圈致动器)



4. When you retrieve(检索) a file from secondary storage and display it on the screen,

a. you are actually retrieving a copy of the desired file(副本) and putting it on the desktop.

b. an old version of the file remains in secondary storage.

c. that file is then sent to ROM.

d. if no file contents is modified after retrieval, the original file will not be replaced when you finish.

ABD

Lecture 22~24

1. **What type of flip-flop(触发器) has an illegal state(非法状态)?**
   SR flip-flop
   (Lecture24 Page10-15)
2. **What type of flip-flop allows us to copy data?**
   D flip-flop
   (Lecture24 Page17-18)
3. **What type of circuit has 'memory'?**
   Sequential logic circuits
   (Lecture24 Page8)