本次作业要求如下：

1.截止日期：2025/05/14 周日晚 24:00

2.**提交作业给本课程 QQ 群里的助教（庄养浩：QQ1473889198）**

3.**命名格式：**附件和邮件命名统一为“第一次作业+学号+姓名”，**作业为 PDF 格式；**

4.注意：

(1) 选择、判断和填空只需要写答案，大题要求有详细过程，**过程算分**。

(2) 答案请用另一种颜色的笔回答，便于批改，**否则视为无效答案。**

(3) 大题的过程最好在纸上写完拍照放 word 里，或者直接在 word 里写。

(4) 本次作业由 Part1、Part2、Part3，需要全部作答

5.本次作业遇到问题请联系课程群里的助教。

Part1 信息的表示和处理+程序优化

一．选择题

1．位于存储器层次结构中的最顶部的是(　)。

A. 寄存器　B. 主存　C. 磁盘　D. 高速缓存

2．关于 Intel 的现代 X86-64 CPU，说法正确的是(　)。

A. 属于 RISC　B. 属于 CISC　C. 属于 MISC　D. 属于 AVX

3．操作系统在管理硬件时使用了几个抽象概念，其中(　)是对处理器、主存和 I/O 设备的抽象表示。

A. 进程　B. 虚拟存储器　C. 文件　D. 虚拟机

4. 以下有关编程语言的叙述中，正确的是(　)。

A. 计算机能直接执行高级语言程序和汇编语言程序

B. 机器语言可以通过汇编过程变成汇编语言

C. 汇编语言比高级语言有更好的可读性

D. 汇编语言和机器语言都与计算机系统结构相关

5. 给定字长的整数 x 和 y 按补码相加，和为 s，则发生正溢出的情况是(　)

A. $x>0,y>0,s\leq 0$ B. $x>0,y<0,s\leq 0$

C. $x>0,y<0,s\geq 0$ D. $x<0,y<0,s\geq 0$

6．设机器数字长 8 位（含 1 位符号位），若机器数 DAH 为补码，分别对其进行算术左移一位和算术右移一位，其结果分别为 (　)

A. B4H, EDH

B. B5H, 6DH

C. B4H, 6DH

D. B5H, EDH

7．–1029 的 16 位补码用十六进制表示为(　)。

A. 8405H
B. 0405H
C. FBFBH
D. 7BFBH

8．已知两个正浮点数，$N_1 = 2^{j1} \times S_1$，$N_2 = 2^{j2} \times S_2$，当下列（）成立时，$N_1 < N_2$。

A. S_1和S_2均为规格化数，且$J_1 < J_2$

B. $S_1 < S_2$

C. S_1和S_2均为规格化数，且$J_1 > J_2$

D. $J_1 < J_2$

9．C 程序执行到整数或浮点变量除以 0 可能发生(　)。
A. 显示除法溢出错直接退出
B. 程序不提示任何错误
C. 可由用户程序确定处理办法
D. 以上都可能

10．补码加法运算的溢出判别中，以下说法正确的是(　)
A. 符号相同的两个数相加必定不会发生溢出
B. 符号不同的两个数相加可能发生溢出
C. 符号相同的两个数相加必定发生溢出
D. 符号不同的两个数相加不可能发生溢出

11．假定变量 i、f 的数据类型分别是 int、float。已知 i=12345，f=1.2345e3，则在一个 32 位机器中执行下列表达式时，结果为“假”的是(　)。
A. i==(int)(float)i
B. i==(int)(double)i
C. f==(float)(int)f
D. f==(float)(double)f

12．以下关系表达式，结果为“真”的是(　)。
A. 2147483647U > –2147483648
B. (unsigned) –1 > –2
C. –1 < 0U
D. 2147483647 < (int) 2147483648U

13．假定某数采用 IEEE 754 单精度浮点数格式表示为 00000001H，则该数的值是（　　）。
A. NaN（非数）
B. 1.0×2^(-149)

C. 1.00…01×2^(-127)
D. 1.0×2^(-150)
14．C 语言程序如下，下列说法叙述正确的是(　)。

```
#include <stdio.h>
#define DELTA sizeof(int)
int main(){
  int i;
  for (i = 40;   i - DELTA >= 0; i -= DELTA)
    printf("%d ",i);
}
```

A. 程序有编译错误
B. 程序输出 10 个数：40 36 32 28 24 20 16 12 8 4 0
C. 程序死循环，不停地输出数值
D. 以上都不对

15．若 int 型变量 x 的最高有效字节全变 0，其余各位不变，则对应 C 语言表达式为(　)。
A. ((unsigned) x << 8) >>8
B. ((unsigned) x >> 8) << 8
C. (x << 8) >>8
D. (x >> 8) << 8

二．填空题

1．64 位系统中 short 数 -2 的机器数二进制表示_____。

2．判断整型变量 n 的位 7 为 1 的 C 语言表达式是_____。

3. -1024 采用 IEEE 754 单精度浮点数格式按内存地址从低到高表示的结果（十六进制表示，小端模式）是_____。

4．C 语言中的 double 类型浮点数用_____位表示。

5．64 位系统中，整型变量 x = -7，其在内存从低地址到高地址依次存放的数是_____(十六进制表示，小端模式)。

三．判断题

1. (　)C 浮点常数 IEEE754 编码的缺省舍入规则是四舍五入。
2. (　)浮点数 IEEE754 标准中，规格化数比非规格化数多。
3. (　)对 unsigned int x，（x*x）>=0 总成立。
4. (　)CPU 无法判断加法运算的和是否溢出。
5. (　)C 语言中的有符号数强制转换成无符号数时位模式不会改变。
6. (　)C 语言中数值从 int 转换成 double 后，数值虽然不会溢出，但是可能是不精确的。
7. (　)C 语言中从 double 转换成 float 时，值可能溢出，但不可能被舍入。

四．分析题

1. 请说明 float 类型编码格式，并按步骤计算 -10.1 的各部分内容，写出 -10.1 在内存从

低地址到高地址的存储字节内容（小端系统）。

2. 向量元素和计算的相关程序如下，请改写或重写计算函数 vector_sum，进行速度优化，并简要说明优化的依据。（如果能自己动手在电脑上测试一下，优化前后性能提升了多少会有额外加分，贴上截图，注明机器型号）

```
/*向量的数据结构定义 */
typedef struct{
        int len;      //向量长度，即元素的个数
        float *data; //向量元素的存储地址
} vec;

/*获取向量长度*/
int vec_length(vec *v){return v->len;}

/* 获取向量中指定下标的元素值，保存在指针参数 val 中*/
int get_vec_element(*vec v, size_t idx, float *val){
      if (idx >= v->len)
        return 0;
      *val = v->data[idx];
      return 1;
}

/*计算向量元素的和*/
void vector_sum(vec *v, float *sum){
      long int i;
      *sum = 0;//初始化为 0
      for (i = 0; i < vec_length(v); i++) {
          float val;
          get_vec_element(v, i, &val);//获取向量 v 中第 i 个元素的值，存入 val 中
          *sum = *sum + val;      //将 val 累加到 sum 中
      }
}
```

Part2 X64 汇编

1.C 语言程序中的整数常量、整数常量表达式是在（　　）阶段初始化和计算的。
（A）预处理　　（B）执行　　（C）链接　　（D）编译
2.关于 Intel 的现代 X86-64 CPU 正确的是（　　）
A. 属于 RISC　B. 属于 MISC　　C. 属于 CISC　　D. 属于 NISC
3.下列叙述正确的是（　　）
A.X86-64 指令"mov $-1,%eax"会使%rax 的值变成 0xffffffffffffffff
B.在一条指令执行期间，CPU 不会两次访问内存

C.CPU 不总是执行 CS::RIP 所指向的指令，例如遇到 call、ret 指令时
D.一条 mov 指令不可以使用两个内存地址操作数
4.在 x86-64 系统中，调用函数 int gt (long x, long y)时，保存参数 y 的寄存器是( )
A. %rdi B.%rsi C.%rax D.%rdx
5.在 x86-64 系统中，调用函数 long gt (long x, long y)时，保存返回值的寄存器是( )
A. %rdi B.%rsi C.%rax D.%rbx
6.下列传送指令中，哪一条是正确的（ ）
A.movb $0x105,(%ebx)
B.movl %rdi,%rax
C.movq %rdi,$105
D.movl 4(%rsp),%eax
7.C 语言程序定义了结构体 struct noname{char c; long n; int k; float *p; short a;};若该程序编译成 64 位可执行程序，则 sizeof(noname)的值是_____。
8.在 X86-64 中，程序的栈存放在内存中，栈顶元素的地址使所有栈中元素中最_____的，栈指针寄存器_____保存着栈顶元素的地址。
9.While 循环的两种展开方法分别是什么：（ ）（ ）
下面代码是哪种？（ ）

```
long fact_while_jm_goto(long n)
{
    long result = 1;
    goto test;
 loop:
    result *= n;
    n = n-1;
 test:
    if (n > 1)
        goto loop;
    return result;
}
```

10. 已知内存和寄存器中的数值情况如下：

内存地址	值
0x100	0xff
0x104	0xAB
0x108	0x13
0x10c	0x11

寄存器	值
%rax	0x100
%rcx	0x1
%rdx	0x3

请填写下表，给出对应操作数的值：

操作数	值
%rax	
(%rax)	

9(%rax,%rdx)	
0xfc(,%rcx,4)	
(%rax,%rdx,4)	

11.有下列 C 函数:

```
long max(long x, long y)
{
    long result;
    if(x >= y){
        result = x;
    }else{
        result = y;
    }
    return result;
}
```

请写出红色部分代码使用条件数据传输来实现条件分支的等价形式,并给出对应的条件传送指令(初始执行指令 movq %rdi, %rax）。

12.假设变量 sp 和 dp 被声明为类型:
Src_t *sp;
Dest_t *dp;
这里的 Src_t 和 Dest_t 是用 typedef 声明的数据类型，我们想使用适当的数据传送指令来实现下面的操作:
*dp = (Dest_t) *sp;
sp 和 dp 的值分别存储在%rdi 和%rsi 中，对下表的每个表项，请写出合适的两条传送指令（如需用到其他寄存器，使用%rax）。注意：规定如果强制类型转换既涉及大小变化又涉及符号变化时，操作应先改变大小。

Src_t	Dest_t	指令
char	int	①_____
char	unsigned	②_____
unsigned char	long	③_____
int	char	④_____
unsigned	unsigned char	⑤_____

13. 简述缓冲区溢出攻击的原理以及防范方法（2 种）

Part3 Y86 处理器体系结构

1. Y86-64 的指令 ret 编码长度为（　　）。

A. 1 字节　　B. 2 字节　　C. 9 字节　　D. 10 字节

2. Y86-64 的 CPU 顺序结构设计与实现中，分成（　　）个阶段

A. 5　　B. 6　　C. 7　　D. 8

3. Y86-64 的 CPU 流水线结构设计与实现中，分成（　　）个阶段

A. 5　　B. 6　　C. 7　　D. 8

判断题：

4. Y86-64 的顺序结构实现中，寄存器文件读时是作为时序逻辑器件看待　（　　）

5. 现代超标量 CPU 指令的平均周期接近于 1 个但大于 1 个时钟周期（　　）

6. 下表是 CPU 的某个场景，解释：加载指令（mrmovq 和 popq）占所有执行指令的 20%，其中 15%会导致 加载/使用 冒险。条件分支指令占所有执行指令的 25%，其中 40%不选择分支。返回指令占所有执行指令的 3%。完成下表：

原因	名称	指令频率	条件频率	气泡数	总处罚	CPI
加载使用	lp	0.20	0.15			
预测错误	mp	0.25	0.40			
返回	rp	0.03	1.00			

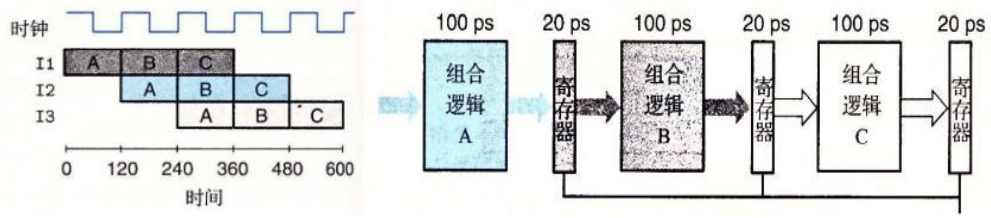
7. 在 Y86-64 架构的机器上，有下列汇编代码，请指出 jne t 指令之后执行的那条指令的地址为_____（顺序执行，不考虑流水线）。

```
0x000:      xorq %rax,%rax
0x002:      jne  t
…
0x019: t:  irmovq $3, %rdx
0x023:      irmovq $4, %rcx
0x02d:      irmovq $5, %rdx
```

8.请写出 Y86-64 的 CPU 流水线结构设计与实现中各流水线阶段的名称（注意顺序不要错位）。

9. Y86-64 流水线 CPU 中的冒险的种类与处理方法。

10. 假设有一个理想的三阶段流水线，执行指令为 I1（Instruction1），I2，I3，其时序图与电路示意图如图所示：（P321）



请分析不同时间点各寄存器中保存值所属指令，并完成下表（填入 I1，I2，I3，None。寄存器 1 表示左数第一个寄存器）

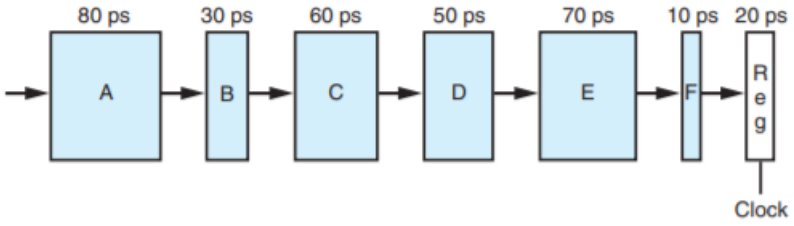
	寄存器 1	寄存器 2	寄存器 3
239			
241			
300			
361			

11. 请根据描述在 Y86-64 顺序实现的条件下完成下面表格。

(1)请写出 Y86-64 顺序实现的 push rA 指令在各阶段的微操作。
(2)请写出 Y86-64 顺序实现的 pop rA 指令在各阶段的微操作。

指令	push rA	pop rA
取指		
译码		
执行		
访存		
写回		
更新 PC		

12.假设我们分析图中的组合逻辑，认为它可以分为 6 个块，以此命名为 A、B、C、D、E、F，延迟分别为 80，30，60，50，70，10（单位 ps），如下图所示。



在这些块之间插入流水线寄存器，就得到这一设计的流水线化的版本。根据在哪里插入流水线寄存器，会出现不同的流水线深度(有多少个阶段)和最大吞吐量的组合。假设每个流水线寄存器的延迟为 20ps。
请根据以上描述，回答下列问题：
(1) 只插入一个寄存器，得到一个两阶段的流水线。要使吞吐量最大化，该在哪里插入寄存器呢?吞吐量和延迟是多少?

(2) 要使一个三阶段的流水线的吞吐量最大化，该将两个寄存器插在哪里呢?吞吐量和延迟是多少?

(3) 要使一个四阶段的流水线的吞吐量最大化，该将三个寄存器插在哪里呢?吞吐量和延迟是多少?

(4) 要得到一个吞吐量最大的设计，至少要有几个阶段?描述这个设计及其吞吐量和延迟。

13.下面是一个程序段，请根据 Y86-64 的微指令和流水线数据相关的知识，试解释为什么在 call 指令之前要插入 3 个 nop 指令。

```
0x000:    irmovq Stack,%rsp    # Intialize stack pointer
0x00a:    nop
0x00b:    nop
0x00c:    nop
0x00d:    call p               # Procedure call
0x016:    irmovq $5,%rsi       # Return point
0x020:    halt
```