秘密★启用前

2021秋ICS小班练习题5
建议用时40分钟

Processor Arch: ISA&Logic
Processor Arch: Sequential

姓名：　　学号：

答卷说明：
a. 答卷前,考生务必将自己的姓名填写在答题卡指定位置.
b. 答题时,请将答案填写在试卷和答题卡相应位置.如需改动,请用签字笔将原答案划去,再在规定位置填写修正后的答案.未在规定区域作答的答案无效.
c. 本卷共7页,卷面分110分.考试结束后,试卷由助教统一收回.

一、选择题(20分)　每题只有一个正确答案

(　　)1. 下面有三组对于指令集的描述，它们分别符合①____，②____，③____ 的特点。
① 某指令集中，只有两条指令能够访问内存。
② 某指令集中，指令的长度都是4字节。
③ 某指令集中，可以只利用一条指令完成字符串的复制，也可以只利用一条指令查找字符串中第一次出现字母K的位置。
A. CISC, CISC, CISC
B. RISC, RISC, CISC
C. RISC, CISC, RISC
D. CISC, RISC, RISC

(　　)2. （注：当年机器是32位的）Y86的指令popl rA的SEQ实现如下图所示，其中1和2分别是：

Fetch	icode:ifun ← M_1[PC] ra:rb ← M_1[PC+1] valP ← ❶
Decode	valA ← R[%esp] valB ← R[%esp]
Execute	valE ← ❷
Memory	valM ← M_4[valA]
Write Back	R[%esp] ← valE R[ra] ← valM
PC Update	PC ← valP

A. PC + 4　　valA + 4
B. PC + 4　　valA + (-4)
C. PC + 2　　valB + 4
D. PC + 2　　valB + (-4)

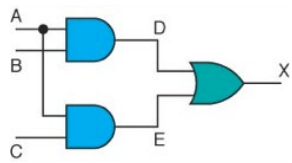
(　　)3. 在 Y86 的 SEQ 实现中，对仅考虑 IRMMOVQ ICALL IPOPQ IRET 指令，对 mem_addr 的 HCL 描述正确的是

```
word mem_addr = [
        icode in { (1), (2) } : valE;
        icode in { (3), (4) } : valA;
];
```

A. (1) IRMMOVQ　　(2) IPOPQ　　(3) IRET　　(4) ICALL

B. (1) IRMMOVQ (2) IRET (3) IPOPQ (4) ICALL
C. (1) ICALL (2) IPOPQ (3) IRMMOVQ (4) IRET
D. (1) IRMMOVQ (2) ICALL (3) IPOPQ (4) IRET

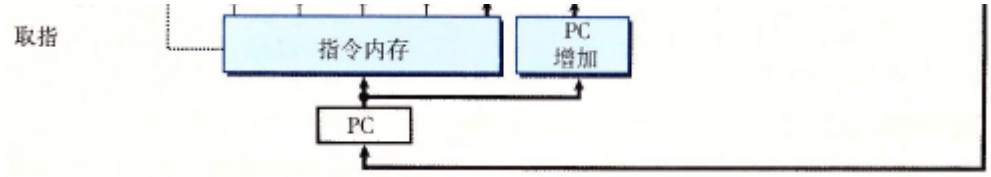
( )4. 对应下述组合电路的正确 HCL 表达式为



A. Bool X = (A || B) && (A || C)
B. Bool X = A || (B && C)
C. Bool X = A && (B || C)
D. Bool X = A || B || C

二、非选择题(90分) 请将答案填写在答题卡上

5. (25分) 请绘制SEQ硬件结构图(不必关注线的粗细).



这是结构图的底部，你需要绘制上面的部分

6.(25分) 请补全Archlab中seq-full.hcl所给出的HCL代码.

```
################################################################################
#    Control Signal Definitions.                                               #
################################################################################
```

每空需填写下面各项中的一个或几个：
IRRMOVQ,IIRMOVQ,IRMMOVQ,IMRMOVQ,IOPQ,IJXX,ICALL,IRET,IPUSHQ,IPOPQ,valA,valB,valC,valP,valM,Cnd.没有提示需要填写几个的空不代表只需要填写一个.

```
################ Fetch Stage     ###################################
# Determine instruction code
word icode = [
        imem_error: INOP;
        1: imem_icode;          # Default: get from instruction memory
];

# Determine instruction function
word ifun = [
        imem_error: FNONE;
        1: imem_ifun;           # Default: get from instruction memory
];

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

# Does fetched instruction require a regid byte?
bool need_regids =
        icode in { (1) ▲ };
```

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

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

## What register should be used as the A source?
word srcA = [
        icode in { IRRMOVQ, IRMMOVQ, IOPQ, IPUSHQ }: rA;
        icode in {IOPQ, IRMMOVQ, IMRMOVQ}: RRSP;
        1 : RNONE; # Don't need register
];

## What register should be used as the B source?
word srcB = [
        icode in { (2) ▲     }: rB;
        icode in { IPUSHQ, IPOPQ, ICALL, IRET }: RRSP;
        1 : RNONE;  # Don't need register
];

## What register should be used as the E destination?
word dstE = [
        icode in { IRRMOVQ } && Cnd : rB;
        icode in { (3) ▲（提示：填两个）  }: rB;
        icode in { IPUSHQ, IPOPQ, ICALL, IRET }: RRSP;
        1 : RNONE;  # Don't write any register
];

## What register should be used as the M destination?
word dstM = [
        icode in {IMRMOVQ, IPOPQ}: rA;
        1 : RNONE;  # Don't write any register
];

################ Execute Stage  ###################################

## Select input A to ALU
word aluA = [
        icode in { IRRMOVQ, IOPQ }: valA;
        icode in { IIRMOVQ, IRMMOVQ, IMRMOVQ }: valC;
        icode in { ICALL, IPUSHQ } : -8;
        icode in { IRET, IPOPQ } : 8;
        # Other instructions don't need ALU
];

## Select input B to ALU
word aluB = [
        icode in { IRMMOVQ, IMRMOVQ, IOPQ, ICALL,
                   IPUSHQ, IRET, IPOPQ }: valB;
        icode in { IRRMOVQ, IIRMOVQ }: 0;
        # Other instructions don't need ALU
];
```

```
## Set the ALU function
word alufun = [
        icode == IOPQ : ifun;
        1 : ALUADD;
];

## Should the condition codes be updated?
bool set_cc = icode in {IOPQ};

################ Memory Stage  ###################################

## Set read control signal
bool mem_read = icode in { IMRMOVQ, IPOPQ, IRET };

## Set write control signal
bool mem_write = icode in { IRMMOVQ, IPUSHQ, ICALL };

## Select memory address
word mem_addr = [
        icode in {———此空不填，答案只比选择第3题多IPUSHQ和IMRMOVQ———}: valE;
        icode in { ———此空不填，答案与选择第3题同———}: valA;
        # Other instructions don't need address
];

## Select memory input data
word mem_data = [
        # Value from register
        icode in { (4) ▲ (提示：填两个)}: valA;
        # Return PC
        icode == ICALL : valP;
        # Default: Don't write anything
];
## Determine instruction status
word Stat = [
        imem_error || dmem_error : SADR;
        !instr_valid: SINS;
        icode == IHALT : SHLT;
        1 : SAOK;
];
################ Program Counter Update ############################
## What address should instruction be fetched at
word new_pc = [
        # Call.  Use instruction constant
        icode == ICALL :ValC;
        # Taken branch.  Use instruction constant
        icode == IJXX && Cnd : valC;
        # Completion of RET instruction.  Use value from stack
        icode == IRET : (5) ▲ ;
        # Default: Use incremented PC
        1 : valP;
];
#/* $end seq-all-hcl */
```

7.(20分)请分析Y86-64 ISA中新加入的一族算术指令： irOpq V, rA, rB，其格式如下：

C	Fn	rA	rB	V(8字节)

与Opq类似，这族指令由四个指令组成，分别是iraddq, irsubq, irandq和irxorq。其功能为：计算R[rA] OP V并将结果存入R[rB]中，这里OP根据Fn的取值分别取+, -, &和^，且此过程会设置条件码寄存器。

若在教材所描述的SEQ处理器上执行这条指令，请按下表补全每个阶段的操作。需说明的信号可能会包括：icode, ifun, rA, rB, valA, valB, valC, valE, valP, Cnd; the register file R[], data memory M[], Program counter PC, condition codes CC。其中对存储器的引用必须标明字节数。如果在某一阶段没有任何操作，请填写none指明。

Stage	irOpq V, rA, rB
Fetch	icode:ifun ← M_1[PC]
Decode	valA ← R[rA] valB ← R[rB]
Execute	
Memory	none
Write Back	
Update PC	PC ← valP

8.(20分)分析32位的Y86 ISA中新加入的条件内存传送指令：crmmovqXX和cmrmovqXX。crmmovqXX和cmrmovqXX指令在条件码满足所需要的约束时，分别执行和rmmovq以及mrmovq同样的语义。其格式如下

rmmovq	4	0	rA	rB	D（8字节）
crmmovqXX	4	fn	rA	rB	D（8字节）
mrmovq	5	0	rA	rB	D（8字节）
cmrmovqXX	5	fn	rA	rB	D（8字节）

请按下表补全每个阶段的操作。需说明的信号可能会包括：icode, ifun, rA, rB, valA, valB, valC, valE, valP, Cnd；寄存器堆R[],存储器M[], 程序计数器PC, 条件码CC。其中对存储器的引用必须标明字节数。

阶段	rmmovq rA,D(rB)	cmrmovqXX D(rB),rA
取指		
译码	valA ← R[rA] valB ← R[rB]	
执行		
访存		
写回	none	
更新PC	PC ← valP	

附HCL描述中的常数值编码表如下：

IHALT	halt指令的代码	INOP	nop指令的代码
IRRMOVQ	rrmovq指令的代码	IIRMOVQ	irmovq指令的代码
IRMMOVQ	rmmovq指令的代码	IMRMOVQ	mrmovq指令的代码
ICRMMOVQ	crmmovqXX指令的代码	ICMRMOVQ	cmrmovqXX指令的代码
IOPL	整数运算指令的代码	IJXX	跳转指令的代码
ICALL	call指令的代码	IRET	ret指令的代码
IPUSHQ	pushq指令的代码	IPOPQ	popq指令的代码
FNONE	默认功能码	RNONE	表示没有寄存器文件访问
ALUADD	表示加法运算	RRSP	表示%rsp寄存器ID
SAOK	正常地址操作状态码	SADR	地址异常状态码
SINS	非法指令异常状态码	SHLT	halt状态码

补充题，不计分(命题人:程业翔)

1.结构体A定义如下:

```
struct A{
int i;
union{
        int a; double d;
};
long l; char s[6];
};
```

在x86-64下，sizeof(A)的运行结果为:

A.32　　B.26　　C.34　　D.30

答案:A

2.下列有几项属于Intel汇编格式与ATT汇编格式的区别?

(1)Intel格式中，寄存器名不需要加'%' 前缀;

(2)Intel格式中，立即数的表示不用带任何前缀;

(3)Intel格式中，目标操作数在源操作数的右边;

A.0　　B.1　　C.2　　D.3

答案:C (3)应为ATT格式中，目标操作数在源操作数的右边，而Intel格式与之相反。

3.当条件码寄存器满足下列哪个条件时，会执行ja跳转命令?

A.~CF&~ZF　　B.~CF　　C.~(SF^OF)&~ZF　　D.~(SF^OF)

答案:A

4.在x86-64下，下列寻址语句正确的有____个.

(1)$8　(2)(%rax, 8)　(3) $8(%rax, %rbx)　(4) (%rax, %rbx, 6)

A.0　　B.1　　C.2　　D.3

答案:B

5.下列说法中，错误的是____.

A.movzlq %eax, %rax 把%eax零扩展到%rax;

B.movslq %eax, %rax 把%eax符号扩展到%rax;

C.cltq 把%eax符号扩展到%rax;

D.cqto 把%rax符号扩展到16个字节，其中%rdx存放前8个字节，%rax存放后8个字节。

答案:A