Chapter 1 引言

1、有两台计算机 A 和 B：A 有乘法指令，而 B 没有；二者都有加法和减法指令；在其余方面，二者都相同。那么，对于 A 和 B，哪台计算机可以解决更多的问题？

所有的计算机（无论大还是小，快还慢，昂贵还是便宜），如果给予足够的时间和足够的存储器，都可以做相同的计算。换句话说，所有的计算机都能做几乎完全相同的事情，只是计算速度上有差别。

2、给出如下问题的算法：

算法是一个逐步计算的过程，该过程一定能够结束，而且每个步骤都能够被明确描述，并能被计算机所执行。

1）计算 1+2+3+4+5+6+7+8+9+10。

解法一：直接运用公式(1+n)n/2，n=10。

解法二：累加求和$S_n = S_{n-1} + a_n, \dots, S_1 = S_0 + a_1, S_0 = 0$，n=10。

2）判定 2010~2500 年中的某一年是不是闰年。

判定公历闰年遵循的一般规律为：四年一闰，百年不闰，四百年再闰。

解法：

```
int isLeapYear(int year)
{
    return (year%4==0)&&(year%100!=0)||(year%400==0);
}
```

3）对一个大于或等于 3 的正整数，判断它是不是质数。

解法：

```
int isPrime(int num)
{
    for (int i=2; i<=sqrt(num); i++){
        if (num%i==0)
            return 0;
    }
    return 1;
}
```

3、当你将计算机升级（如更换 CPU）后，原来的软件（如操作系统）还能够使用吗？

能用理由：计算机升级升的是硬件，软件存储在磁盘中，只要升级的硬件提供与原有硬件相同的工作方式和功能，软件还是能够正常工作的。

不能用理由：计算机中的硬件升级前与升级后所需要的驱动可能不一样，软件无法通过原有方法调用底层硬件，因此也就不能使用了。

4、你购买的软件通常是以什么方式存在的？是高级语言还是目标机器 ISA 兼容的机器语言？

软件可能存在的方式是多种多样的：源代码或目标代码。源代码包括汇编语言、3GL 和 4GL、经验知识等。目标代码包括机器语言、解释型源代码等。

具体可参考：http://www.rogerclarke.com/SOS/PaperLiaby.html

5、对计算系统的每个抽象层次，请分别举出 2 个以上的例子。

计算系统的抽象层次：问题、算法、程序、操作系统、指令集系统、微处理器、逻辑电

路、元件。

6、你对计算系统哪一部分比较熟悉？熟悉程度如何？

开放题

Chapter 2 C 程序设计简介

1、对于如下算法：

i. 从键盘获取 A

ii. X ← A+1

iii. Y ← X+A

iv. Z ← Y-A

v. 输出 Z 到屏幕上

(1) 使用解释技术将其翻译为机器语言，至少需要执行多少次算术运算？

3 次（X ← A+1，Y ← X+A，Z ← Y-A）

(2) 使用编译技术，在将其翻译为机器语言之前，对这段代码进行优化，那么，至少需要执行多少次算术运算？

2 次（X ← A+1，Y ← X+A，Z ← X）

2、如下语句的输出分别是什么？

printf (“%d\n%d\n”, 12, 12 + 45);　　12[换行]57[换行]

printf (“%d,%d\n”, 12, 12 + 45);　　12,57[换行]

printf (“%d %d\n”, 12, 12 + 45);　　12 57[换行]

printf (“%d%d\n”, 12, 12 + 45);　　1257[换行]

printf (“%d.%d\n”, 12, 12 + 45);　　12.57[换行]

Chapter 3 类型和变量

3.1 假设 a 和 b 都是整数，且 a 和 b 分别被赋值为 7 和 8，那么，下列表达式的值分别是多少？并且，如果 a 和 b 的值发生变化，还需要指出 a 和 b 的新值。

1）a = b	8，a=8,b=8
2）a = b =5	5，a=5,b=5
3）a % b	7，a=7,b=8
4）b % a	1，a=7,b=8
5）a || b	1，a=7,b=8
6）a && b	1，a=7,b=8
7）! (a/b)	1，a=7,b=8
8）++a + b- -	16，a=8,b=7
9）a = b+=1	9，a=9,b=9
10）a= (b++==8)? a:b	7，a=7,b=9
11）a= (++b==8)? a:b	9，a=9,b=9

3.2 假设现在新设计出一个计算机程序设计语言，包括运算符+，-，*和/，在下列不同的限定条件下，表达式 a+b-c*d/e 的计算结果分别是什么（使用圆括号表示运算顺序）？

1）优先级顺序为：*和/优先级相同，+和-优先级相同，且*和/的优先级高于+和-，结合性均为自左至右；

((a+b)-((c*d)/e))

2）优先级顺序为：*和/优先级相同，+和-优先级相同，且*和/的优先级高于+和-，结合性均为自右至左；

(a+(b-(c*(d/e))))

3）优先级顺序为：*和/优先级相同，+和-优先级相同，且*和/的优先级低于+和-，结合性均为自左至右；

((((a+b)-c)*d)/e)

4）优先级顺序为：*和/优先级相同，+和-优先级相同，且*和/的优先级低于+和-，结合性均为自右至左；

((a+(b-c))*(d/e))

5）优先级顺序为：+，-，*，/的优先级是依次降低的，即+优先级最高，/的优先级最低；

((((a+b)-c)*d)/e)

6）优先级顺序为：+，-，*，/的优先级是依次升高的，即+优先级最低，/的优先级最高；

(a+(b-(c*(d/e))))

7）四个运算符的优先级相同，结合性均为自左至右；

((((a+b)-c)*d)/e)

8）四个运算符的优先级相同，结合性均为自右至左。

(a+(b-(c*(d/e))))

3.3 假设新设计出一个程序设计语言，赋值运算符具有最高的优先级，其他均与 C 语言完全相同。

1）如下语句的结果是什么？即执行该语句后，i 的值是什么？

i = 2 * 3;　　答：i 的值为 6

2）如何改变这条语句，使得它可以实现相同的 C 语句的作用？

i=6 或 i=(2*3)

3.4 假设一个程序包含两个整数变量 a 和 b，分别有值 7 和 8。请写出可以交换 a 和 b 的值的 C 语句，即执行这些语句后，a 和 b 的值分别是 8 和 7。

与上一题不同的是，题目要求可以交换 a 和 b 的值，而不是实现相同的作用，所以不能像 i=6 那样直接赋值。

1）可以使用一个临时的变量，即第三个变量，写出这些 C 语句；

c=a; a=b; b=c

2）不使用临时变量（即，只有 a 和 b 两个变量），写出这些 C 语句。

解法一：a=a+b; b=a-b; a=a-b;

解法二：a=a^b; b=a^b; a=a^b;

Chapter 4 结构化程序设计和控制结构

4.1 当 x 等于 0、1 和 2 时，下列代码片段的输出各是什么？

1） if (1>=x>=0)

printf (“True. ”);

else

printf ("False. ");

答：x=0 时 True. x=1 时 True. x=2 时 True.

2） if (1>=x && x>=0)

printf (“True. ”);

else

printf ("False. ");

答：x=0 时 True. x=1 时 True. x=2 时 False.

3） if (x=0)

printf ("x equals 0\n");

else if(x=1)

printf ("x equals 1\n");

else

printf ("x does not equal 0 or 1\n ");

答：x=0 时 x equals 1[换行]

x=1 时 x equals 1[换行]

x=2 时 x equals 1[换行]

4） if (x==0)

printf ("x equals 0\n");

else if(x==1)

printf ("x equals 1\n");

else

printf ("x does not equal 0 or 1\n ");

答：x=0 时 x equals 0[换行]

x=1 时 x equals 1[换行]

x=2 时 x does not equal 0 or 1 [换行]

5）switch (x) {

case 0:

printf ("x equals 0\n");

case 1:

printf ("x equals 1\n");

break;

default:

printf ("x does not equal 0 or 1\n ");

break;

}

答：x=0 时 x equals 0[换行] x equals 1[换行]

x=1 时 x equals 1[换行]

x=2 时 x does not equal 0 or 1 [换行]

4.2 下列代码片段的输出是什么？

1）
```
int i;
int sum=0;
for(i=1;i<=50;i++)
    if(i%7==0)
        sum=sum+i;
printf("%d\n ",sum);
```
答：196（7+14+21+28+35+42+49=7*(1+7)*7/2=196）

2）
```
int i;
int sum=0;
for(i=1;i<=50;i+=2)
    if(i%7==0)
        sum=sum+i;
printf("%d\n ",sum);
```
答：112（7+21+35+49=7*(1+7)*4/2=112）

3）
```
int i=10;
while (i>0) {
    i--;
}
printf ("%d ", i);
```
答：0

4）
```
int i=1;
int sum=0;
do{
    if(i%7==0)
        sum=sum+i;
    i++;
}while(sum<100);
printf("%d\n",sum);
```
答：105（7+14+21+28+35=7*(1+5)*5/2=105）

5）
```
int i=10;
do {
    i--;
} while (i>0);
printf ("%d ", i);
```
答：0

4.3 如下代码的输出是什么？

```
int input;
int i;
int j;
int sum=0;
scanf ("%d", &input);

for (i = 1; i <= input; i++) {
    for (j = 0; j < i; j++) {
        sum += j;
    }
}
printf ("sum = %d\n", sum);
```

答：sum=286（1+(1+2)+…+(1+2+…+(n-1))=(n+5)(n+2)(n+1)/12）

4.4 如下代码的输出是什么？

1）

```
int i;
int j;
for (i=4; i>=1; i--){
    for (j=1; j<=i; j++)
        printf ("#");
    for (j=1; j<=4-i; j++)
        printf ("*");
        printf ("\n");
}
```

答：####[换行]###*[换行]##**[换行]#***[换行]

2）

```
int i;
for(i=1;i<=5;i++)
    switch(i%5){
    case 0:
        printf(“*”); break;
    case 1:
        printf(“#”); break;
    default:
        printf(“\n”);
    case 2:
        printf(“&”);
    }
```

答：#&[换行]& [换行]&*

4.5 改写如下代码片段，要求不使用 continue 语句，仍然实现这段代码的功能。

```
for (i=1; i<10; i++){
    if (i%2==0)
        continue;
    printf ("%d ", i);
}
```

解法一：

```
for (i=1; i<10; i++){
    if (i%2==0)
        ;
    else
        printf ("%d ", i);
}
```

解法二：

```
for (i=1; i<10; i++){
    if (i%2!=0)
        printf ("%d ", i);
}
```

Chapter 6 数据的机器级表示

6.1 分别写出 19 和-19 的二进制原码、反码和补码表示（使用 8 位二进制数位）。

答：对于 19，原码 0001 0011，反码 0001 0011，补码 0001 0011；

对于-19，原码 1001 0011，反码 1110 1100，补码 1110 1101。

6.2 将下列二进制数转化为十进制数，假设此二进制数分别为原码、反码和补码整数。

1）0111

答：若是原码，则为 7；若是反码，则为 7；若是补码，则为 7。

2）1110

答：若是原码，则为-6；若是反码，则为-1；若是补码，则为-2。

3）11111111

答：若是原码，则为-127；若是反码，则为 0；若是补码，则为-1；

4）10000000

答：若是原码，则为-0；若是反码，则为-127；若是补码，则为-128；

6.3 将下列十进制数分别转化为 8 位二进制原码、反码和补码整数。

1）-86

答：原码 1101 0110，反码 1010 1001，补码 1010 1010。

2）85

答：原码 0101 0101，反码 0101 0101，补码 0101 0101。

3）-127

答：原码 1111 1111，反码 1000 0000，补码 1000 0001。

4）127

答：原码 0111 1111，反码 0111 1111，补码 0111 1111。

6.4 做下列二进制补码整数加法运算，给出十进制形式的结果，并判断是否产生溢出。

1）1101+01010101

答：1111 1101+0101 0101=(1)0101 0010=82，没有溢出。

2）0111+0101

答：0111+ 0101=1101=-3，溢出。

3）11111111+01

答：1111 1111+0000 0001=(1)0000 0000=0，没有溢出。

4）01+1110

答：0001+ 1110=1111=-1，没有溢出。

5）0111+0001

答：0111+0001=1000=-8，溢出。

6）1000+11

答：1000+1111=(1)0111=7，溢出。

7）1100+00110011

答：1111 1100+0011 0011=(1)0010 1111=47，没有溢出。

8）1010+101

答：1010+1101=(1)0111=7，溢出。

6.5 做下列二进制数计算，结果以二进制形式给出：

1）01010111 AND 11010111　　0101 0111

2）(0011 AND 0110) AND 1101　　0000

3）0011 AND (0110 AND 1101)　　0000

4）01010111 OR 11010111　　1101 0111

5）(0011 OR 0110) OR 1101　　1111

6）0011 OR (0110 OR 1101)　　1111

7）NOT (1011) OR NOT (1100)　　0111

8）NOT (1000 AND (1100 OR 0101))　　0111

9）NOT (NOT (1101))　　1101

10）(0110 OR 0000) AND 1111　　0110

6.6 请给出下列十进制数的 IEEE 浮点数表示：

1）3.75

答：11.11=1.111*2^(128-127)=0 10000000 11100000000000000000000。

2） $-55\frac{23}{64}$

答：-110111.010111=-1.10111010111*2^(128+4-127)

=1 10000100 10111010111000000000000。

3）3.1415927

答：11.001001=1.1001001*2^(128-127)=0 10000000 10010010000000000000000。

6.7 请给出下列 IEEE 浮点数的十进制数表示：

1）0 10000000 00000000000000000000000

答：1*2^(128-127)=2

2）1 10000011 00010000000000000000000

答：-1.0001*2^(128+3-127)=-10001=-17

3）1 10000000 10010000000000000000000

答：-1.1001*2^(128-127)=-11.001=-3.125

6.8 请将下列二进制补码整数的十六进制表示转换为十进制数：

1）xF0

答：1111 0000=-16

2）x7FF

答：0111 1111 1111=2047

3）x16

答：0001 0110=22

4）x8000

答：1000 0000 0000 0000=-2^15=-32768

6.9 请将下列十进制数转换为二进制补码整数的十六进制表示：

1）256

答：0001 0000 0000=x100

2）111

答：0110 1111=x6F

3）-44

答：1101 0100=xD4

6.10 请分别给出下列数的十六进制表示：

1）675.625 的 IEEE 754 浮点数

答：1010100011.101=1.010100011101*2^(128+8-127)

=0 10001000 01010001110100000000000

=4428e800

2）ASCII 字符串：Hello

答：48 65 6c 6c 6f

6.11 如下代码分别输出哪些内容？

1）printf ("%c\n", 13 + 'A'); N[换行]

2）printf ("%x\n", 13); d[换行]

6.12 解释如下代码段的作用：

```
scanf ("%c", &nextChar);
printf ("%d\n", nextChar);
```

答：从键盘接收一个字符，再以十进制的方式打印到屏幕上。

6.13 请描述如下代码段的作用及输出：

```
int i;
scanf ("%d", &i);
for (j=0; j<16; j++) {
    if (i & (1 << j)) {
        count++;
    }
}
printf ("%d\n", count);
```

答：从键盘接收一个整数，统计其二进制形式中 1 的位数，并打印到屏幕上。

6.14 请给出如下代码段的输出：

```
int x=20;
int y=10;
while ((x > 10 ) && (y & 15 )) {
        y=y+1;
        x=x-1;
        printf ("*");
}
```

答：******

Chapter 7 数字逻辑电路

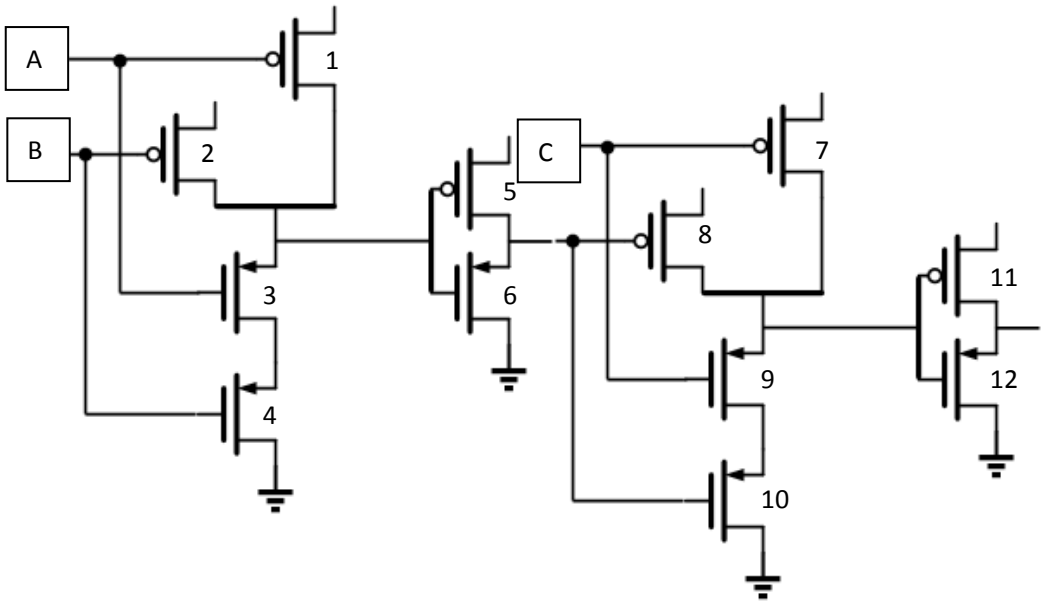
7.1

1）请画出三个输入的与门的晶体管级电路图。

2）对于如下输入，在其晶体管级电路图中标出其表现。

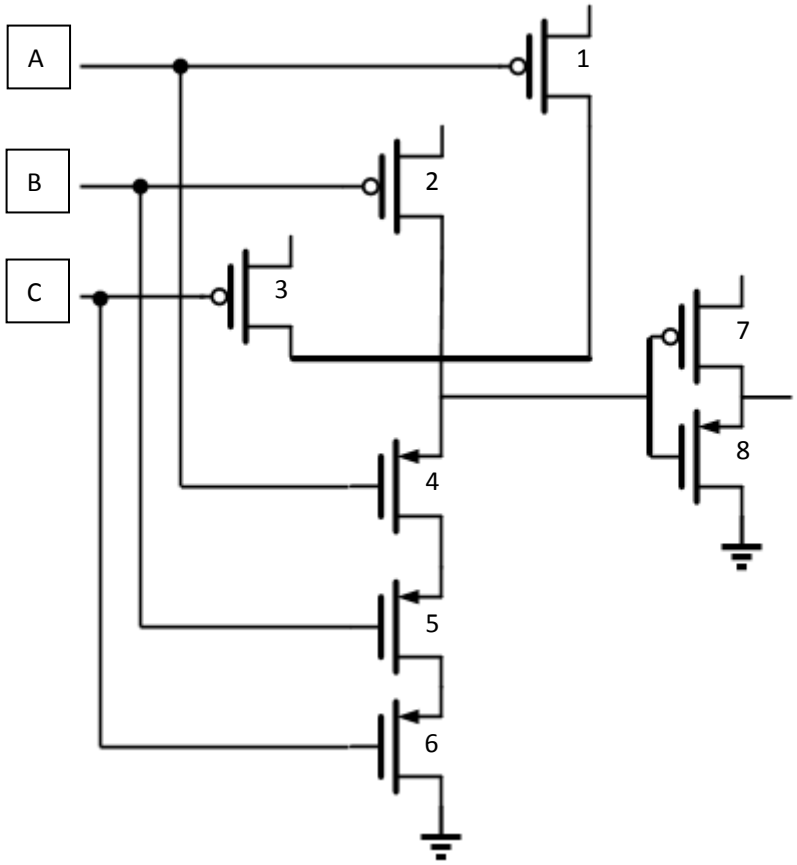
A=0, B=0, C=1

解法一：



当 A=0，B=0，C=1 时，1/2/6/8/9/12 晶体管连通，最终输出 0。

解法二：

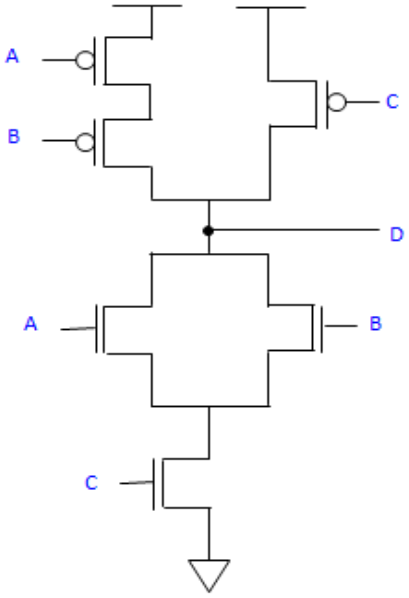


当 A=0，B=0，C=1 时，1/2/6/8 晶体管连通，最终输出 0。

7.2

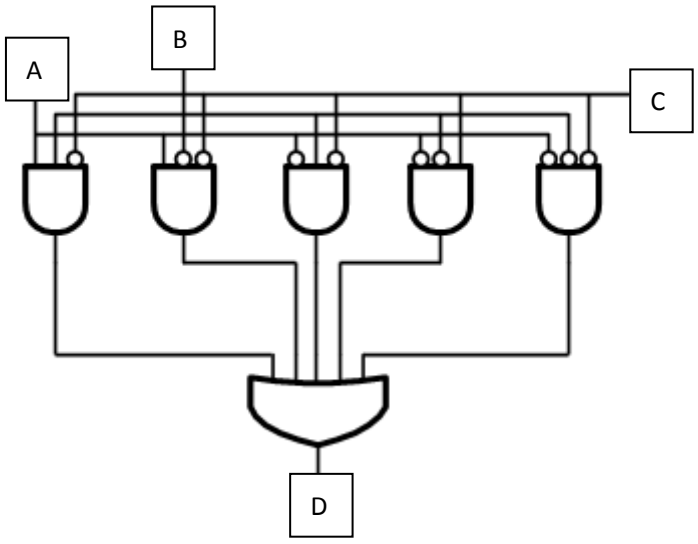
1）给出下图所示的晶体管级电路的真值表。

2）使用与、或、非门，给出该真值表的门级电路图。



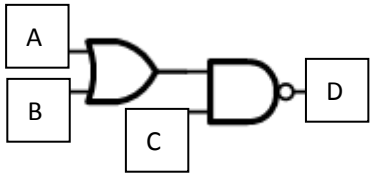
A	B	C	D
1	1	1	0
1	1	0	1
1	0	1	0
0	1	1	0
1	0	0	1
0	1	0	1
0	0	1	1
0	0	0	1

解法一：直接绘制门级电路图。

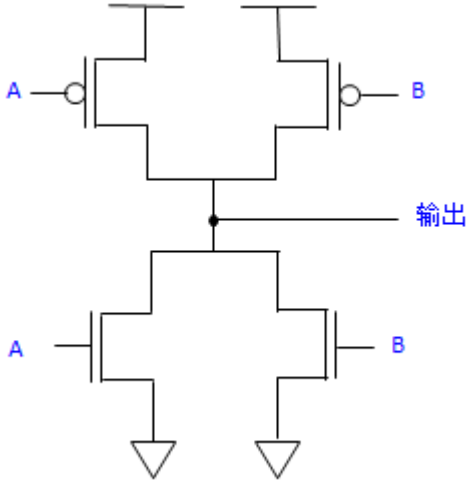


解法二：化简后绘制门级电路图。

$$
\begin{aligned}
D &= AB\bar{C} + A\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}\bar{B}C + \bar{A}\bar{B}\bar{C} \\
&= A\bar{C} + \bar{A}\bar{C} + \bar{A}\bar{B}C \\
&= \bar{C} + \bar{A}\bar{B} \\
&= \overline{C(A+B)}
\end{aligned}
$$

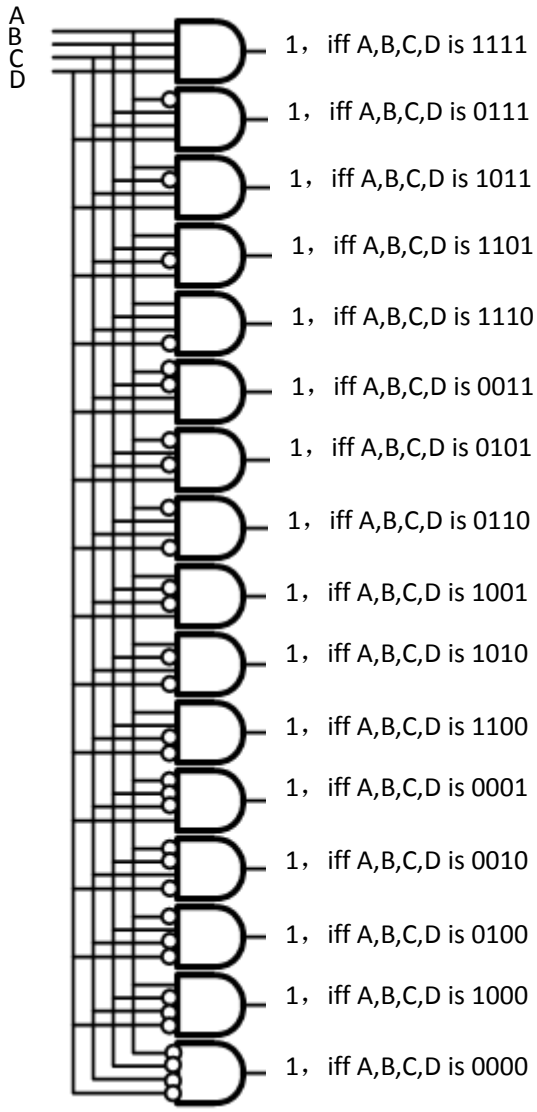


7.3 如下图所示的电路有一个缺陷，请指出该缺陷。

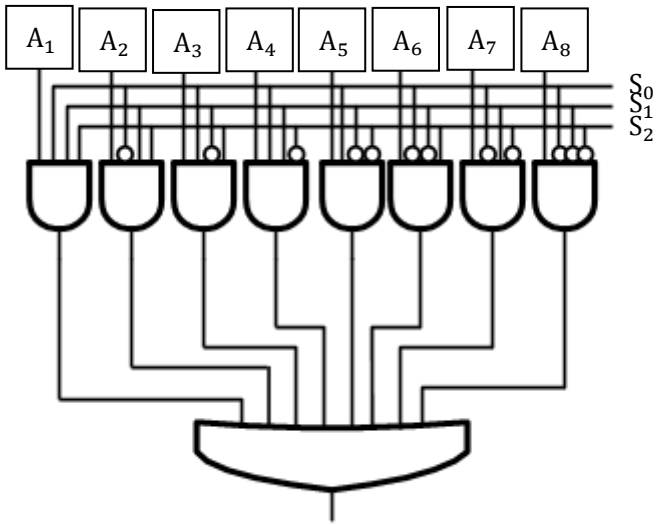


答：若 A=0,B=1 或 A=1,B=0，则同时接到了电源正极和大地负极，不能判断输出结果。

7.4 请画出有 4 个输入的译码器的门极电路图，并注明各输出为 1 的条件。



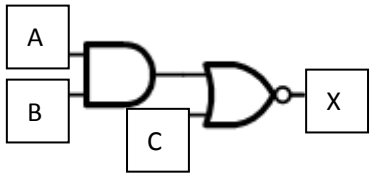
7.5 请画出有 8 个输入的多路选择器的门极电路图。



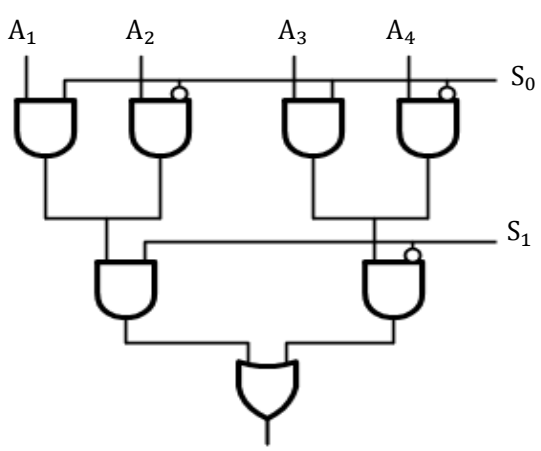
7.6 对于如下真值表，请使用 7.3.4 节（可编程逻辑阵列）给出的算法，生成其门级逻辑电路。

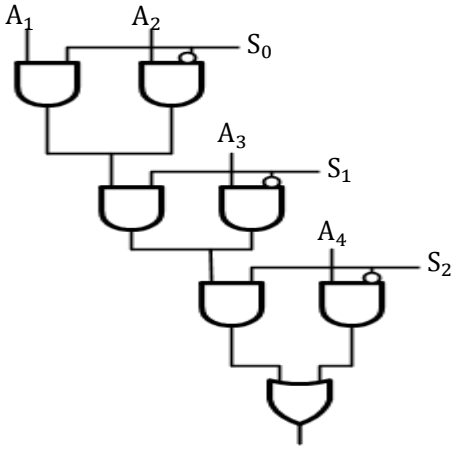
A	B	C	X
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

解：$X = \overline{A}\overline{B}\overline{C} + \overline{A}B\overline{C} + A\overline{B}\overline{C} = (\overline{A} + A\overline{B})\overline{C} = \overline{AB + C}$

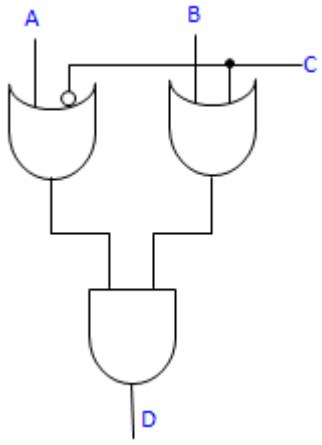


7.7 只使用 2 选 1 的多路选择器，就可以实现 4 选 1 的多路选择器，给出其电路图。





7.8 根据下图所示的逻辑电路图，写出相应的真值表。



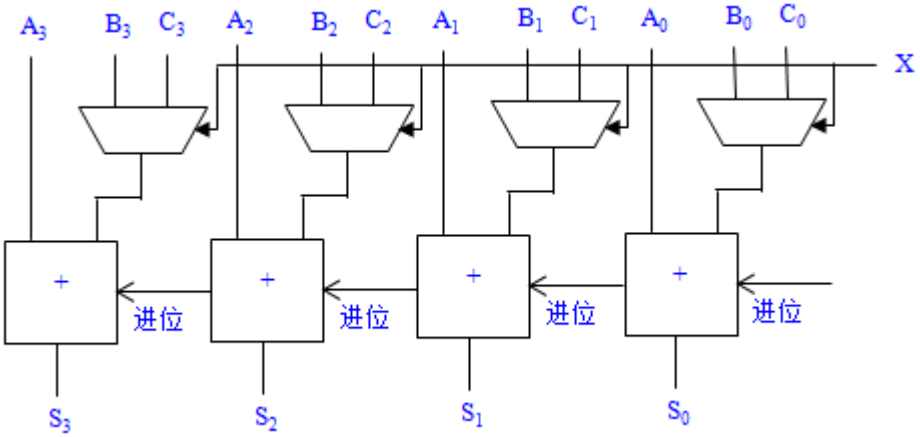
A	B	C	D
1	1	1	1
1	1	0	1
1	0	1	1
0	1	1	0
1	0	0	0
0	1	0	1
0	0	1	0
0	0	0	0

7.9

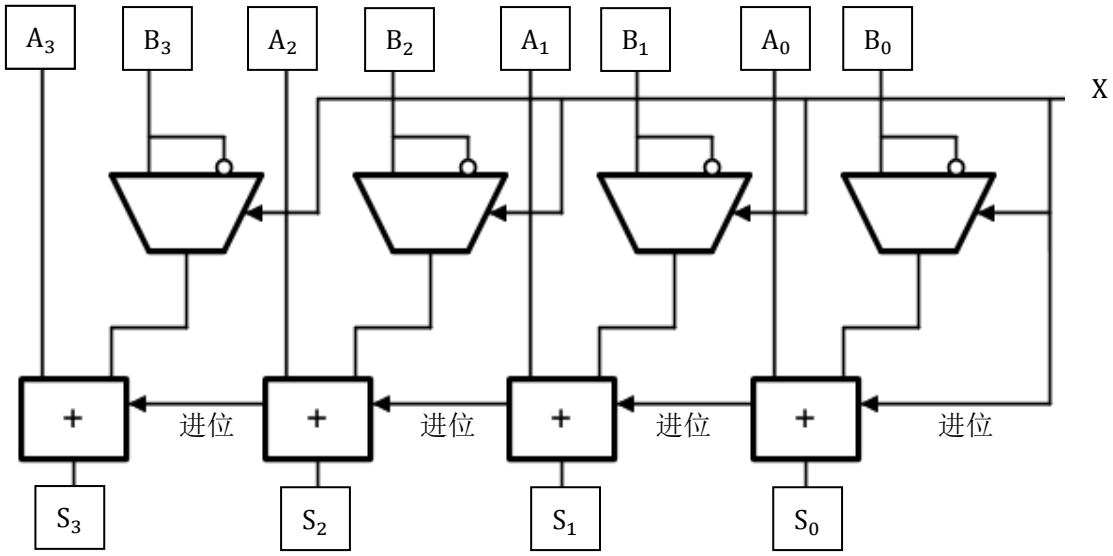
1）下图中的每个矩形都表示一个全加法器，当 X=0 和 X=1 时，电路的输出分别是什么？

答：分别为$S_i = A_i + B_i +$ 进位，$S_i = A_i + C_i +$ 进位，$i = 0, 1, 2, 3$。

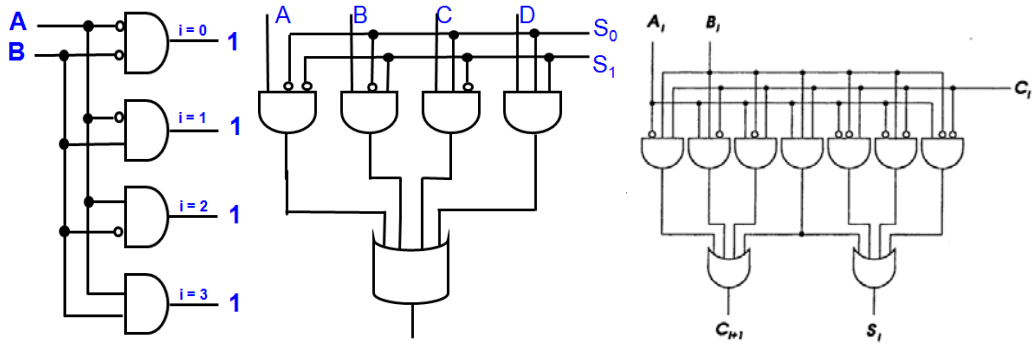
2）在该电路图的基础上，构建一个可以实现加法/减法运算的逻辑电路图。即，取决于 X 的值，电路计算 A+B 或 A-B 的值。



答：当 X=0 时，计算 A+B；当 X=1 时，计算 A-B（理由是——取反加一）。



7.10 一个逻辑结构的速度与从输入到达输出，需传递经过的逻辑门的最长路径有关。假设与、或、非门都被计为一个门延迟，例如，两个输入的译码器的传递延迟等于 2（参照图 7.14），这是因为有些输出需经过两个门的传递。



1）两个输入的多路选择器的传递延迟是多少（参照图 7.15）？

答：3

2）1 位的全加法器的传递延迟是多少（参照图 7.18）？

答：3

3）4 位的全加法器的传递延迟是多少（参照图 7.19）？

答：4*3=12

4）32 位的全加法器的传递延迟是多少？

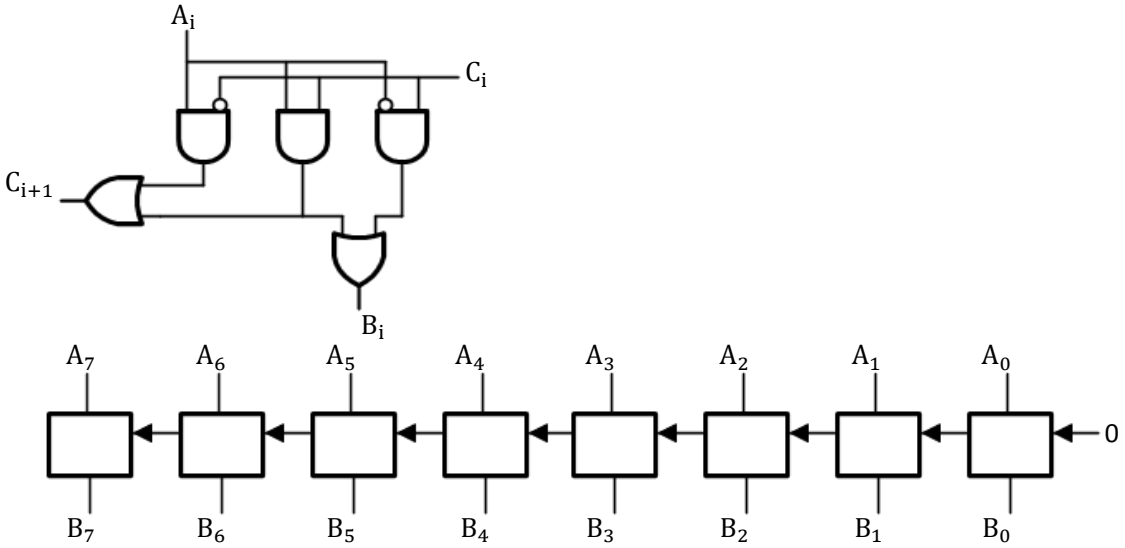
答：32*3=96

7.11 设计一个 8 位的移位器，该移位器的输入和输出分别是 A[7:0]和 B[7:0]。B[7:0]是 A[7:0]向左移动 1 位的结果。

1）写出此移位器的真值表。

A_i	C_i	B_i	C_{i+1}
1	1	1	1
1	0	0	1
0	1	1	0
0	0	0	0

2）使用与、或和非门实现此移位器。

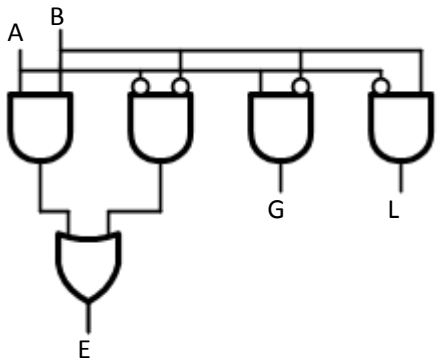


7.12 设计一个 1 位的比较器，该比较器电路有两个 1 位的输入 A 和 B，有 3 个 1 位的输出 G（greater，大于 、E（equal，等于）和 L（less，小于）。当 A > B 时，G 为 1，否则，G 为 0；当 A = B 时，E 为 1，否则，E 为 0；当 A < B 时，L 为 1，否则，L 为 0。

1）给出此 1 位比较器的真值表。

A	B	G	E	L
1	1	0	1	0
1	0	1	0	0
0	1	0	0	1
0	0	0	1	0

2）使用与、或、非门实现此比较器电路。



7.13 参照下图，回答问题：

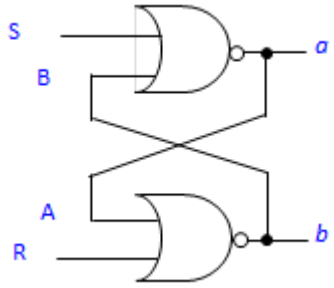
1）当 S 和 R 都为 0 时，此逻辑电路的输出是什么？

答：取决于组成门的晶体管的电子特性而不是取决于被操作的逻辑值。

2）如果 S 从 0 转换到 1，输出是什么？

答：0（a 的值为 0；假如 a 的值为 1，则 b 为 0，则 a 为 0，所以 a 只能为 0）

3）此逻辑电路是存储元件吗？



答：是

7.14 某个计算机有 4 个字节的寻址能力，访问其存储器的一个单元需要 64 位，该存储器的大小是多少（以字节为单位）？此存储器共存储多少位？

答：2^64*4=2^66(byte)，2^66*8=2^69(bit)。

7.15 8 位被称为一个字节（byte），4 位被称为一个单元组（nibble）。一个字节可寻址的存储器使用 14 位的地址，那么，此存储器共存储了多少单元组？

答：2^14*8/4=2^15(nibble)。

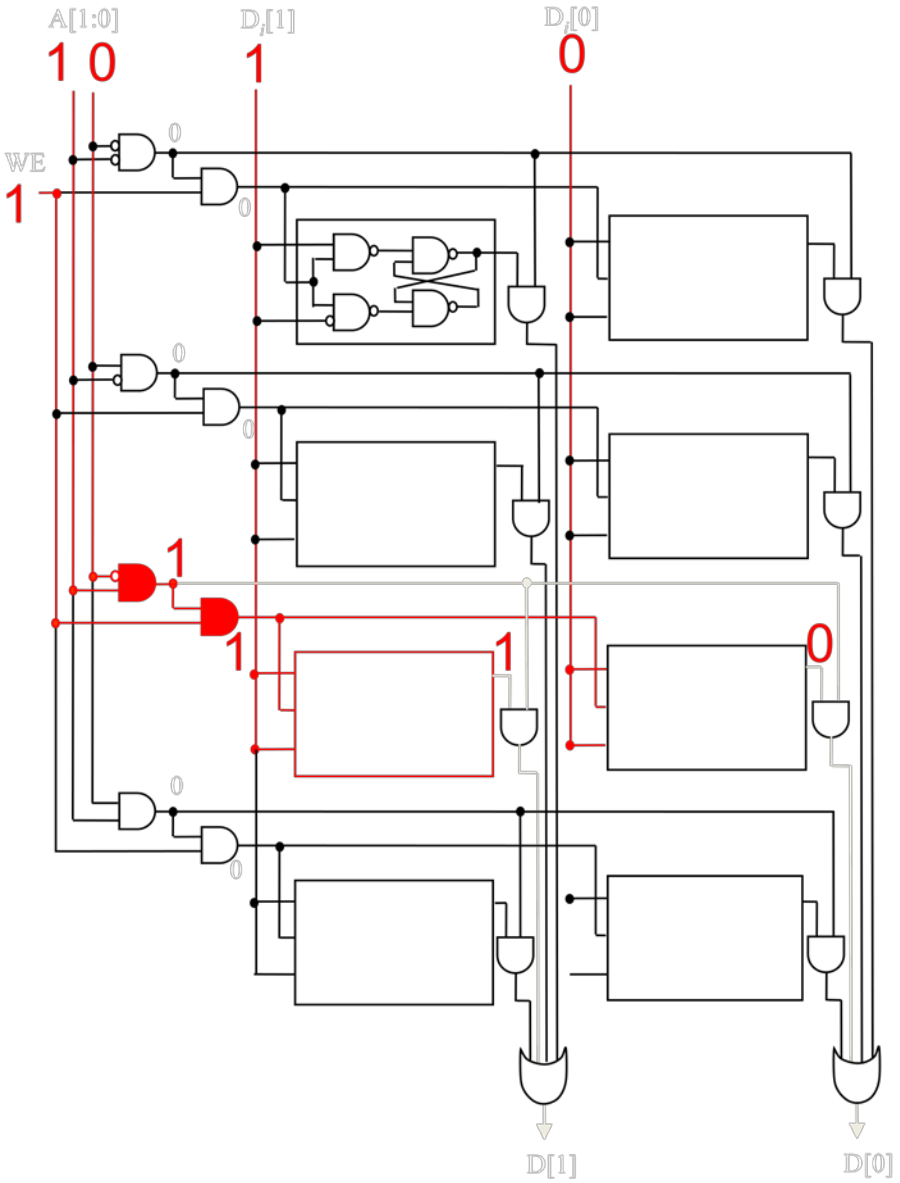
7.16 对于图 7.24 所示的 4×2 位大小的存储器，回答以下问题：

1）如果向单元 3 存储数值 10，A[1:0]和 WE 必须被设置为什么值？

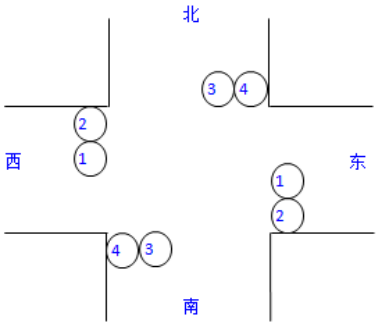
答：A[1:0]必须被设置为 10 或 11（有没有单元 0？），WE 必须被设置为 1。

2）如果将此存储器的单元数目从 4 增长到 10，需要多少条地址线？存储器的寻址能力是否发生变化？

答：需要 4 条地址线，存储器的寻址能力不变。（寻址能力是指每个单元的位数）

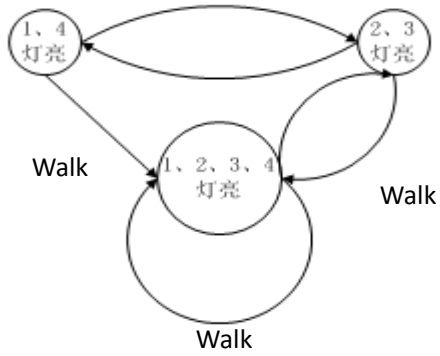


7.17（选做题）设计一个简单的交通灯控制器。与 7.6.4 节的时序逻辑电路类似，系统的背景是东西向大街（EW）和南北向大街（NS）相交的十字路口。控制器的输入是 Walk 按钮，由希望过马路的行人按下。输出是两个信号 EW 和 NS，分别控制 EW 和 NS 方向的红绿灯。当 EW=1，NS=0 时，东西向大街的绿灯亮，而南北向大街的红灯亮；当 EW=0，NS=1 时，东西向大街的红灯亮，而南北向大街的绿灯亮。当没有行人时，EW=1，NS=0 保持 1 分钟；然后，EW=0，NS=1 保持 1 分钟。这个过程循环进行，EW 和 NS 交替变化。当有行人按下 Walk 按钮，在当前的 1 分钟结束时，EW 和 NS 都变为 1，并保持 1 分钟（东西向大街和南北向大街的绿灯亮），然后，回到 EW=1，NS=0，继续交替变化。



（红灯 1、3，绿灯 2、4）

1）画出状态图。



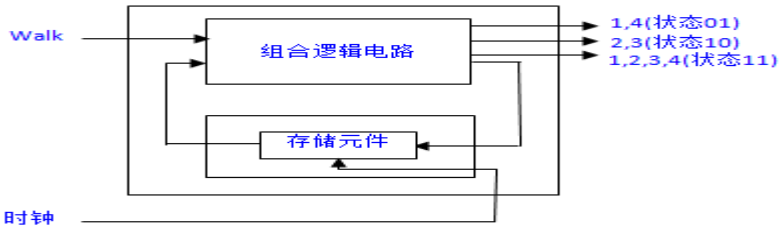
2）写出输出函数表和下一状态函数表。

答： 定义状态 01 为 1/4 灯亮，状态 10 为 2/3 灯亮，状态 11 为 1/2/3/4 灯亮。

Walk	Status	1/4	2/3	1/2/3/4
0	1/4	0	1	0
1	1/4	0	0	1
0	2/3	1	0	0
1	2/3	0	0	1
0	1/2/3/4	0	1	0
1	1/2/3/4	0	0	1

Walk	Current Status	Next Status
0	01	10
1	01	11
0	10	01
1	10	11
0	11	10
1	11	11

3）给出此交通灯控制器的时序逻辑电路。



Chapter 8 冯·诺依曼(Von Neumann)模型

8.1 下表显示了一个小的存储器，请根据此表回答问题：

地址	数据
00000000	00000000
00000001	11111110
00000010	10000000
00000011	01111111
00000100	01000010
00000101	11010101
00000110	10000000
00000111	00000000
00001000	00000000
00001001	01100100
00001010	00101000
00001011	00000001
.....	.....

1）单元 0 和单元 4 包含的二进制数值分别是什么？

答： 单元 0 包含$(00000000)_2 = (0)_{10}$；

单元 4 包含$(01000010)_2 = (66)_{10}$。

2）每个单元内的二进制数值可以以不同的方式被解释，如可以表示为无符号整数、补码整数、浮点数、ASCII 码等，

I. 将单元 0 和单元 1 解释为 8 位补码整数，请以十进制形式写出结果；

答：补码整数下，单元 0 表示 0，单元 1 表示-2。

II. 将单元 2 和单元 3 解释为 8 位无符号整数，请以十进制形式写出结果；

答：无符号整数下，单元 2 表示 128，单元 3 表示 127。

III. 将单元 4 解释为 ASCII 码值；

答：ASCII 码值下，单元 4 为$(01000010)_2 = (42)_{16}$，查表可知其为字母 B。

IV. 将单元 4、5、6 和 7 解释为一个 IEEE 浮点数（32 位），其中，单元 4 包含该数的[31:24]位，单元 5 包含[23:16]位，单元 6 包含[15:8]位，单元 7 包含[7:0]位，请以十进制形式写出结果。

答：0 10000101 10101011000000000000000=1.10101011*2^(128+5-127)

=1101010.11=106.75

3）存储单元的内容也可以是一条指令，将单元 8、9、10 和 11 解释为一条指令，其中，单元 8 包含该指令的[31:24]位，单元 9 包含[23:16]位，单元 10 包含[15:8]位，单元 11 包含[7:0]位，该指令表示什么？

答：000000 00011 00100 00101 00000 000001

bit[31:26]：000000——R 类型

bit[5:0]：要执行的函数，000001，是加法操作。

bit[15:11]：存储的结果所在的位置，R5。

bit[25:21]和 bit[20:16]：存储两个源操作数的寄存器，R3 和 R4。

因此，该指令表示将 R3（寄存器 3）和 R4 里的内容相加，结果存回 R5 里。

4）一个二进制数值也可以被解释为一个存储单元的地址，如果存储在单元 11 中的数值是一个地址，它指的是哪个单元？那个单元里包含的二进制数值是什么？

答：单元 11 中存储的是 00000001，即单元 1；

那个单元中的二进制数值为$(111111110)_2 = (-2)_{10}$。

Chapter 9 指令集结构

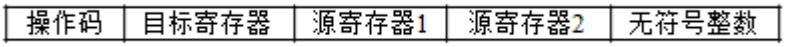
9.1 假设一个 16 位的指令采取如下格式：

如果共有 12 个操作码和 8 个寄存器，那么，“补码整数”能够表示的数值范围是什么？

答：12 个操作码需要 4 位表示，8 个寄存器需要 3 位表示；

所以补码整数最多有 6 位，能够表示的数值范围是$-2^5 \sim 2^5 - 1$。

9.2 假设一个 32 位的指令采取如下格式：

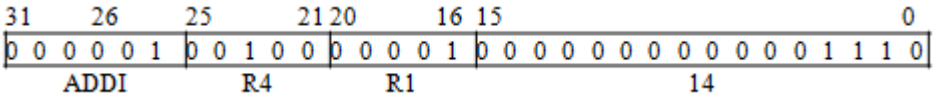


如果共有 200 个操作码和 60 个寄存器，“无符号整数”能够表示的最大数是多少？

答：200 个操作码需要 8 位表示，60 个寄存器需要 6 位表示；

所以无符号整数最多有 6 位，能够表示的最大数是$2^6 - 1 = 63$。

9.3 对于 DLX 的 I-类型指令，请回答以下问题：



1）立即数的范围是多少？

答：I-类型指令的第二个源操作数来自于指令[15:0]进行符号扩展得到的 32 位整数；

共有 16 位，所以其表示范围为$-2^{15} \sim 2^{15} - 1$。

2）如果重新定义 DLX 的 ISA，使得立即数表示无符号整数，那么，立即数的范围是多少？

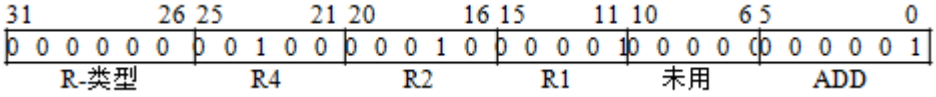
答：若重新定义立即数表示无符号整数，可假设进行无符号扩展得到 32 位整数，否则会出现表示范围的不连续性。从而，立即数的表示范围变为$0 \sim 2^{16} - 1$。

3）如果重新定义 DLX 的 ISA，将寄存器的数量从 32 个降低到 16 个，那么，在 I-类型的指令中能够表示的立即数的最大值是多少？假设立即数仍表示补码整数。

答：将寄存器的数量从 32 个降低到 16 个，所需位数减一；

所以补码整数最多有 18 位，能够表示的最大值是$2^{17} - 1$。

9.4 对于 DLX 的 R-类型指令，如果重新定义 DLX 的 ISA，将寄存器的数量从 32 个增加至 128 个，是否可行？



答：在 R-类型指令中，有 5 个位没有使用；

将寄存器的数量从 32 个增加到 128 个，所需位数加 2；

总共需要位数加 6，而只有 5 个位可用，故而不可行。

9.5 假设某计算机的存储器包括 65536 个单元，每个单元包含 16 位的内容，请回答以下问题：

1）需要多少位表示地址？

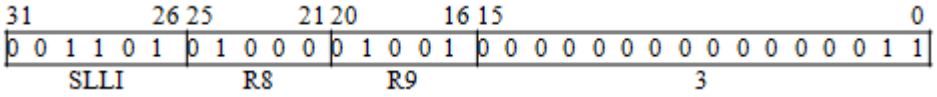
答：2^16=65536，故需要 16 位表示地址。

2）假设每条指令都由 16 位组成，其中一条指令与 DLX 无条件跳转指令（J 指令）工作机制类似。如果该指令位于单元 10 中，要跳转至地址 20，那么 PC 相对偏移量应为多少？

答：20-(10+1)=9。

9.6 将 R8 乘以 8，并将结果存于 R9 中，一条 DLX 指令（图 9.4 给出的指令）可以实现吗？

答：如果可以，相当于将 R8 左移 3 位后存入 R9 中。



如果不可以，说明理由，比如左移 3 位可能导致高位溢出的问题。

9.7 执行 DLX 指令“ADD R10，R8，R9”之后，发现 R8[31]等于 R9[31]，但是不等于 R10[31]。已知 R8 和 R9 包含的是无符号整数，那么，在何种条件下，R10 中的结果是可信的？

答：R8[31]等于 R9[31]，但是不等于 R10[31]；

若 R8[31]为 1，则 R8、R9 的数值范围是2^{31}~$2^{32}-1$，R10 最大值为$2^{31}-1$，不可信；

若 R8[31]为 0，则 R8、R9 的数值范围是2^{30}~$2^{31}-1$，R10 如果是补码整数，其值为负不可信，R10 如果是无符号整数，其数值范围为2^{31}~$2^{32}-1$，涵盖了 R8+R9 的范围2^{31}~$2^{32}-2$，所以结果是可信的。

9.8

1）使用一条 DLX 指令，可以将 R1 中的值移至 R2 中吗？

答：可以——ORI R2, R1, #0；ANDI R2, R1, #-1；ADDI R2, R1, #0；ADD R2, R1, R0。

2）使用一条 DLX 指令，可以将 R1 中的值按位取反吗？

答：可以——XORI r1, r1, #-1

3）假设 R1 中存储的位组合的最右边两位有特殊的重要性，根据这两位的数值，处理 4 个任务之一。使用一条 DLX 指令，将这两位孤立出来。

答：ANDI R1, R1, #2

9.9 当一段起始于单元 x30000000 的程序结束执行后，R1~R6 的值分别是多少？

地址	数据	解释
x3000 0000	001100 00000 00001 0100 0000 0000 0000	LHI R1, 0x4000
x3000 0004	011100 00001 00010 0000 0000 0000 0100	LW R2, 4(R1)
x3000 0008	010110 00001 00011 0000 0000 0000 0100	LB R3, 4(R1)
x3000 000C	011101 00001 00010 0000 0000 0000 0000	SW 0(R1), R2
x3000 0010	011100 00010 00100 0000 0000 0000 0100	LW R4, 4(R2)
x3000 0014	010110 00001 00101 0000 0000 0000 0001	LB R5, 1(R1)
x3000 0018	010111 00010 00100 0000 0000 0000 0011	SB 3(R2), R4
x3000 001C	011100 00001 00110 0000 0000 0000 0000	LW R6, 0(R1)
x3000 0020	011100 00110 00110 0000 0000 0000 0000	LW R6, 0(R6)
x3000 0024	110000 00000 00000 0000 0000 0000 0000	TRAP 0
.....	.....	.....
.....	.....	.....
x4000 0000	1000 0111 0110 0101 0100 0011 0010 0001	0x87654321
x4000 0004	0100 0011 0010 0001 0000 0000 0000 0000	0x43210000
.....	.....	.....
.....	.....	.....
x4321 0000	0000 0000 0000 0000 0100 0000 0000 0000	0x00004000
x4321 0004	0000 0000 0000 0000 1000 0000 0000 0000	0x00008000

Refer: Opcode

00=R_is, 01=addi, 03=subi, 09=andi,

0A=ori, 0B=xori, 0C=lhi, 0D=slli, 0E=srli, 0F=srai,

10=slti, 12=slei, 14=seqi, 16=lb, 17=sb,

1C=lw, 1D=sw, 28=beqz, 29=bnez, 2C=j, 2D=jr, 2E=jal, 2F=jalr,

30=trap, 31=RFE

答： R1=0x4000 0000;

R2=M[4+R1]=M[0x4000 0004]=0x4321 0000;

R3=SextM[4+R1]=SextM[0x4000 0004]=Sext(0x43)=0x0000 0043;

M[0x4000 0000]=M[R1]=R2=0x4321 0000;

R4=M[4+R2]=M[0x4321 0004]=0x0000 8000;

R5=SextM[1+R1]=SextM[0x4000 0001]=Sext(0x21)=0x0000 0021;

M[0x4321 0003]=M[3+R2]=R4_27…31=0x00; M[0x4321 0000]=0x0000 4000;

R6=M[R1]=M[0x4000 0000]=0x4321 0000;

R6=M[R6]=M[0x4321 0000]=0x0000 4000;

R1=0x4000 0000，R2=0x4321 0000，R3=0x0000 0043，

R4=0x0000 8000，R5=0x0000 0021，R6=0x0000 4000。

9.10 下表显示了 DLX 存储器的一部分情况：

地址	数据
x3000 0000	001011 00001 00001 1111 1111 1111 1111
x3000 0004	000000 00001 00010 00011 00000 000001
x3000 0008	001011 00011 00011 1111 1111 1111 1111
x3000 000C	101000 00011 00000 1111 1111 1111 0000

如果条件分支将控制转移到 x3000 0000 单元，那么 R1 和 R2 有什么特点？

答：xori r1, r1, #-1; r1=~r1;

add r3, r1, r2; r3=r1+r2;

xori r3, r3#-1; r3=~r3;

beqz r3, 0xFFF0; 因此 r3=0。

r3=~(~r1+r2)=0, ~r1+r2=-1, -r2=~r1+1=~r2+1, r1=r2。

9.11 如果在如下 DLX 指令序列执行结束时，R1 中存储的值为 7，由此可推知 R2 的什么信息？

地址	数据	解释
x3000 0000	000001 00000 00001 0000 0000 0000 0000	ADDI R1, R0, #0
x3000 0004	000001 00000 00011 0000 0000 0001 0000	ADDI R3, R0, 0x10
x3000 0008	000001 00000 00100 0000 0000 0000 0001	ADDI R4, R0, #1
x3000 000C	000000 00010 00100 00101 00000 001001	AND R5, R2, R4
x3000 0010	101000 00101 00000 0000 0000 0000 0100	BEQZ R5, 0x04
x3000 0014	000001 00001 00001 0000 0000 0000 0001	ADDI R1, R1, #1
x3000 0018	001101 00100 00100 0000 0000 0000 0001	SLLI R4, R4, #1
x3000 001C	000011 00011 00011 0000 0000 0000 0001	SUBI R3, R3, #1
x3000 0020	101001 00011 00000 1111 1111 1110 1000	BNEZ R3, 0xFFE8
x3000 0024	110000 00000 00000 0000 0000 0000 0000	TRAP 0

答：R1=0;

R3=16;

R4=1;

Do{

R5=R2&R4;

If (R5!=0)

R1=R1+1;

R4=R4*2;

R3=R3-1;

}while(R3!=0);

此程序检测 R2 的二进制形式中 1 的个数，

因此，R1 为 7 可以知道 R2 的二进制形式中有 7 个 1 和 16-7=9 个 0。