UKÁZKY PROGRAMOVÝCH ŘEŠENÍ LOGICKÉHO KOMBINAČNÍHO OBVODU

Pro připomenutí možností jazyka C si ukážeme různá řešení programu realizující funkci logického kombinačního obvodu se 4 vstupy v proměnné vstup = 00000…00KZYZ a jedním výstupem = 00000….00000F.

$$F = X.Y + X.K + Z.K.\overline{Y}$$

S klasickými logickými obvody můžeme funkci realizovat:

- Obvody NAND, NOR, OR, AND, AND-OR-INVERT, Reed-Mule-rovou formou ⇒
  Programové řešení bude vycházet z realizace logického výrazu

- Multiplexerem, Dekodérem, Pamětí ⇒
  Programové řešení bude založené na realizaci pravdivostní tabulky

- **Programové řešení logického výrazu**
  1. **Realizované posuny a logickými operacemi**

```
Příklad 1.1
unsigned int vstup;  // vstup=000…0000KZYX
unsigned int výstup; // výstup=000…000000F
unsigned int soucin1, soucin2

void main() {
   soucin1=vstup&(vstup>>1);   // 0bit=X.Y, 2bit=Z.K
   soucin2=vstup&(vstup>>3);   // 0bit=X.K
   vystup=((soucin1&(~vstup<<1))>>2); // 0bit=Z.K.negY
   vystup=(vystup|soucin1|soucin2)&0x01; // 0bit=F
                        PROČ NÉ
   vystup=(vystup||soucin1||soucin2)&0x01; // 0bit=F
Stop: goto stop;
}
```

Příklad 1.2

```
void main()
{
        if ((vstup&0x01)&&(vstup&0x02)) goto out;
        if ((vstup&0x01)&&(vstup&0x08)) goto out;
        if ((vstup&0x08)&&(vstup&0x04)&&
            (!(vstup&0x02))) goto out;
konec:  goto konec;
out:    vystup=1; goto konec;
};
```

2. Realizované logickými operacemi mezi bitovými proměnnými

```
typedef struct jednotlive_bity
{       unsigned X : 1;
        unsigned Y : 1;
        unsigned Z : 1;
        unsigned K : 1;
} bit;

void main()
{    if ((vstup&0x01)!=0) bit.X=1;
     if ((vstup&0x02)!=0) bit.Y=1;
     if ((vstup&0x04)!=0) bit.Z=1;
     if ((vstup&0x08)!=0) bit.K=1;
     vystup=(bit.X&&bit.Y)||(bit.X&&bit.K)||
            (bit.Z&&bit.K&&~bit.Y);        // Vystup F
stop: goto stop;
}
```

```
Příklad 2.2
Union
{ unsigned int napln_bit_pole;
      struct jednotlive_bity
      {  unsigned X : 1;
         unsigned Y : 1;
         unsigned Z : 1;
         unsigned K : 1;
       } bit_pole;
}  prenos;

void main()
{    prenos.napln_bit_pole = vstup;
     vystup=(prenos.bit_pole.X && prenos.bit_pole.Y)||
            (prenos.bit_pole.X && prenos.bit_pole.K)||
            (prenos.bit_pole.Z && prenos.bit_pole.K &&
            (~prenos.bit_pole.Y)); // Vystup F
}
```

Příklad 2.3

```
void main()
{  vystup=(1&((vstup&(vstup>>1))|
              (vstup&(vstup>>3))|
              ((vstup>>2)&(vstup>>3)&(!(vstup>>1)))));
};
```

Příklad 2 pro procesory 8051

```
bdata unsigned char vstup; // Globální proměnná
                           // bitově i bytově dostupná
sbit K = vstup^7;          // Definice bitů v proměnné
sbit Z = vstup^6;          // vstup
sbit Y = vstup^5;
sbit X = vstup^4;
sbit F = P1^2;       // Výstup přímo na vývod brány

main() {
opak: vstup=P2;      // přečtení vstupních proměnných
      F=(X&&Y)||(X&&K)||(Z&&K&&~Y);    // Vystup F
      goto opak;
}
```

- **Programové řešení založené na realizaci pravdivostní tabulky**

1. **Realizované příkazem IF**
2. **Realizované příkazem IF-ELSE**
3. **Realizované příkazem SWITCH**
4. **Realizované polem v datové paměti**
5. **Realizované polem v programové paměti**
6. **Realizované komprimovaným polem**

1.1 Realizované příkazem IF

```
unsigned int vstup;  // vstup=000…0000KZYX
unsigned int výstup; // výstup=000…000000F

void main()
{       vystup=0;
        if (vstup==3) vystup=1;
        if (vstup==7) vystup=1;
        if (vstup==9) vystup=1;
        if (vstup==11) vystup=1;
        if (vstup==12) vystup=1;
        if (vstup==13) vystup=1;
        if (vstup==15) vystup=1;
konec: goto konec;
}
```

1.2 Realizované příkazem IF

```
unsigned int vstup;  // vstup=000…0000KZYX
unsigned int výstup; // výstup=000…000000F

void main()
{       vystup=0;
        if (vstup==3) goto out;
        if (vstup==7) goto out;
        if (vstup==9) goto out;
        if (vstup==11) goto out;
        if (vstup==12) goto out;
        if (vstup==13) goto out;
        if (vstup==15) goto out;
konec: goto konec;
out:    vystup=1; goto konec;
}
```

2.1 Realizované příkazem IF nebo IF-ELSE

```
unsigned int vstup;  // vstup=000…0000KZYX
unsigned int výstup; // výstup=000…000000F

void main()
{    if ((vstup==3)||(vstup==7)||(vstup==9)||
        (vstup==11)||(vstup==12)||(vstup==13)||
        (vstup==15)) vystup=1; else vystup=0;
konec:  goto konec;
}
```

2.2 Realizované příkazem IF-ELSE

```
unsigned int vstup;  // vstup=000…0000KZYX
unsigned int výstup; // výstup=000…000000F

void main()
{   vystup=0;
    if (vstup==3) vystup=1;
    else {if (vstup==7) vystup=1;
       else {if (vstup==9) vystup=1;
          else {if (vstup==11) vystup=1;
             else {if (vstup==12) vystup=1;
                else {if (vstup==13) vystup=1;
                   else {if (vstup==15) vystup=1; }}}}}}
}
```

3. Realizované příkazem SWITCH

```
unsigned int vstup;  // vstup=000…0000KZYX
unsigned int výstup; // výstup=000…000000F

void main()
{    vystup=0; vstup&=0x0F;
     switch( vstup )
     {    case 3: vystup=1; break;
          case 7: vystup=1; break;
          case 9: vystup=1; break;
          case 11: vystup=1; break;
          case 12: vystup=1; break;
          case 13: vystup=1; break;
          case 15: vystup=1; break;
      }
}
```

4. Realizované polem v datové paměti

```
unsigned int vstup;  // vstup=000…0000KZYX
unsigned int výstup; // výstup=000…000000F
unsigned int polefunkce[16]={0,0,0,1,0,0,0,1,0,1,0,
                             1,1,1,0,1};

void main()
{       vystup=0;
        vstup&=0x0F;          // Ošetření vstupní hodnoty
        vystup=polefunkce[vstup];
konec:  goto konec;
}
```

5. Realizované polem v programové paměti

```
unsigned int vstup;  // vstup=000…0000KZYX
unsigned int výstup; // výstup=000…000000F
static unsigned int  polefunkce[16]={0,0,0,1,0,0,0,
                                     1,0,1,0,1,1,1,0,1};

void main()
{       vystup=0;
        vstup&=0x0F;     // Ošetření vstupní hodnoty
        vystup=polefunkce[vstup];
konec: goto konec;
}
```

6. Realizované komprimovaným polem

```
unsigned int vstup;  // vstup=000…0000KZYX
unsigned int výstup; // výstup=000…000000F
unsigned int funkce; // Mohla by být i v programové
                     // paměti

void main()
{       vystup=0;
        funkce=0x115D;
        vstup&=0x0F;      // Ošetření vstupní hodnoty
        vystup=(funkce>>vstup)&0x01;
konec: goto konec;
}
```

Programové řešení logického výrazu

Program s prázdnou smyčkou main();
Code=414 RO-data=462 RW-data=4 ZI-data=1028

1.1 - Logické operace a posuny proměnné vstup

Code=498 RO-data=462 RW-data=20 ZI-data=1028

1.2 - Logické operace a posuny proměnné vstup

Code=506 RO-data=462 RW-data=12 ZI-data=1028

1.3 - Logické operace a posuny proměnné vstup
Code=482 RO-data=462 RW-data=12 ZI-data=1028

2.1 - Bitové pole
Code=606 RO-data=462 RW-data=16 ZI-data=1024

2.2 - Union s naplněním bitového pole
Code=518 RO-data=462 RW-data=16 ZI-data=1024

Proč je tak velký rozdíl mezi 2.1 a 2.2 ?

VYHODNOCENÍ PROGRAMOVÝCH ŘEŠENÍ LKO

- **Řešení založené na realizaci pravdivostní tabulky**

1.1 - Operace IF s nastavením výstupu
Code=526 RO-data=462 RW-data=12 ZI-data=1028

1.2 - Operace IF se skokem na nastavením výstupu
Code=506 RO-data=462 RW-data=12 ZI-data=1028

2.1 - Realizované příkazem IF nebo IF-ELSE
Code=494 RO-data=462 RW-data=12 ZI-data=1028

2.2 - Realizované příkazem IF-ELSE
Code=538 RO-data=462 RW-data=12 ZI-data=1028

3. Realizované příkazem SWITCH
Code=526 RO-data=462 RW-data=12 ZI-data=1028

4. Realizované polem v datové paměti
Code=504 RO-data=524 RW-data=12 ZI-data=1028

5. Realizované polem v programové paměti
Code=458 RO-data=462 RW-data=76 ZI-data=1028

6. Realizované komprimovaným polem
Code=470/462 RO-data=462 RW-data=16 ZI-data=1024