6) Lågnivåprogrammering – påverka enskilda bitar

(Ex) Hårdvara på ett kort avbildas i minnet som 0100**1**0001

Denna bit återställer hårdvaran till startläge om den sätts till 0!

(Ex)

```
unsigned char c1, c2, c3, c4
c1 = 'A';      // Hela byten
c2 = 5;        //  — " —
c3 = c1+c2;    //  — " —
c4 = c1 | c2;  // Enskilda bitar
```

c1	01000001	65
c2	00000101	5
c3	01000110	70
c4	01000101	

Bitoperatorer

Vänsterskift	c1 = c1 << 5;	c1	00100000
Högersskift	c2 = c2 >> 2;	c2	00000001
Bitvis or	c3 = c1 | c2;	c3	00100001
Bitvis and	c4 = c2 & c3;	c4	00000001
Bitvis xor	c5 = c3 ^ c4;	c5	00100000
Bitvis not	c6 = ~c5;	c6	11011111

Prioritet! Se Bilting sid 63

↑ ~
*/
+ -
<< >>
< >
==
&
|
↓ =

(Ex) a) Är bit 5 satt?

```
                Bit5   Bit0
c1          0010 0000
1           0000 0001
Mask 1<<5   0010 0000
c1 & 1<<5   0010 0000
```

```
if ((c1 & (1<<5)) == 0)
{
    printf("Ej satt!");
}
else
{
    printf("Satt!");
```

b) Sätt bit nr 3

```
                Bit3
c1          0010 0000
Mask 1<<3   0000 1000
c1 | 1<<3    0010 1000
```

```
c1 = c1 | (1<<3);
```

OBS! Oberoende av om biten är 0 eller 1 vid start!

OBS! Påverkan ej andra bitar

c) Nolla bit nr5

```
c1          0010 1000
Mask 1<<5   0010 0000
~(1<<5)     1101 1111
c1 & (~(1<<5))  0000 1000
```

```
c1 = c1 & (~(1<<5));
```

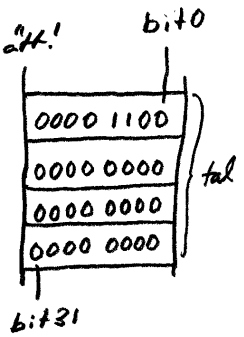
d) Toggla (1→0, 0→1) bit nr 2

```
c1          0000 1000
1<<2        0000 0100
c1 ^ 1<<2   0000 1100
```

```
c1 = c ^ (1<<2);
```

Testa även 1 ?

Utmatning – i minnet finns bara ettor och nollor men de kan tolkas på olika sätt!



(Ex)

```
int tal = 12;
printf("Tal decimalt : %d", tal);
// Tal decimalt : 12
printf("Tal hexadecimalt : %x", tal);
// Tal hexadecimalt : C
printf("Tal oktalt : %o ", tal);
// Tal oktalt : 14
```

Binärt? Finns ej i C!

(Ex) Skriv en funktion som skriver ut ett heltal binärt enligt: 0000 0000 0000 0000 0000 0000 0000 1100
(bit 31 ↑ bit 0 ↑)

```
void skriv_bin(int tal)
{
    int i;

    for (i=31; i>=0; i--)
    {
        if ((tal & (1<<i)) == 0)
        {
            putchar('0');
        }
        else
        {
            putchar('1');
        }
    }
}
```

(Ex) Digsims tal-sk med bitoperatorer

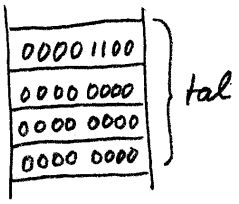
```
void tal-sk (int tal, char *str, int nrbits)
{
   int i, j=0;

   for (i=nrbits-1; i>=0; i--)
   {
      if ((tal & (1<<i)) ==0)
      {
         str[j] = '0';
      }
      else
      {
         str[j] = '1';
      }
      j++;
   }
}
```

Inmatning

(Ex)

```
int tal;
printf("Ge decimalt tal: ");
scanf("%d", &tal);
// Ge decimalt tal: 12
printf("Ge hexadecimalt tal: ");
scanf("%x", &tal);
// Ge hexadecimalt tal: C
```

Binärt? Finns ej i C



(Ex) skriv en funktion som läser ett heltal binärt.

```
int las_bin(void)
{
   int i, tal=0;
   printf("Ge binärt tal: ");
   for(i=31; i>=0; i--)
   {
      if(getchar()=='1')
      {
         tal = tal | (1<<i);
      }
   }
   return tal;
}
```

(Ex) Digsums sh-tal med bitoperatorer

```
int sh_tal(char *str)
{
   int i, j=0, tal=0;
   for(i=shlen(sh)-1; i>=0; i--)
   {
      if (str[j]=='1')
         tal = tal | (1<<i);
      j++;
   }
   return tal;
}
```

(Ex) Packa ihop talen 5 och 9 i en byte där varje tal tar upp 4bitar

```
unsigned char twonr;
```

Packa:
```
twonr = 5;          00000101
twonr = twonr<<4;   01010000
twonr = twonr|9;    01011001
                    (5)(9)
```

Packa upp:
```
Maska 0xf  00001111
twonr & 0xf ger 9
twonr >> 4  ger 5
```

Bitstructar – ger namn åt bitarna

(Ex)

```
bs  0100 0001
    ↑        ↑
   bit7     bit0
```

```
struct byte
{
   unsigned int bit0 : 1;   <- Anger antalet bitar för termen
   unsigned int bit1 : 1;
      |
   unsigned int bit7 : 1;
} bs = {1,0,0,0,0,0,1,0};
        ↑               ↑
       bit0            bit7
```

(Ex) a) Testa om bit 5 är satt i bs.

```
if (bs.bit5)
{  printf("Satt!");
}
else
{  printf("Ej satt!");
}
```

b) Sätt bit 3

```
bs.bit3 = 1;
```

c) Nolla bit 5

```
bs.bit5 = 0;
```

d) Toggla bit 2

```
bs.bit2 = !bs.bit2;
```

Har man en given variabel tilldelar man adressen till en bitstruct pekare

(EX) Antag att du har en given variabel
unsigned char uch = 'A'. Vi namnsätter bitarna med

```
struct byte
{
    unsigned int bit0:1;
    unsigned int bit1:1;
    :
    unsigned int bit7:1;
} *bsp = (struct byte *) &uch; //obs! Typomvandling
```

a) Testa om bit 5 är satt i uch

```
if (bsp->bit5)
{   printf("Satt!");
}
else
{   printf("Ej satt!");
}
```

b) Sätt bit 3 i uch

```
bsp->bit3 = 1;
```

c) Nolla bit 5 i uch

```
bsp->bit5 = 0;
```

d) Toggla bit 2 i uch

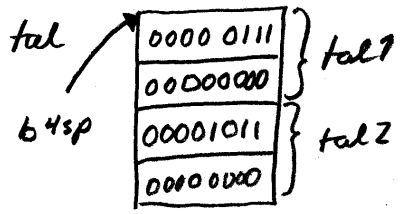
```
bsp->bit2 = !bsp->bit2;
```

(Ex) Spara minne genom att packa ihop data i 4bitarsenligt:

twonr 0101 1001
5 9
tal2 tal1

```
struct byte
{
    unsigned int tal1 : 4;
    unsigned int tal2 : 4;
} twonr;

unsigned int summa, temp;
twonr.tal1 = 9;
twonr.tal2 = 5;
summa = twonr.tal1 + twonr.tal2;
printf("Summan = %u\n", summa);

/*Läs in nya tal till tal1 och tal2 */
printf("Ge ett tal mellan 0 och 15 : ");
scanf("%u", &temp);  //OBS! Kan ej läsa direkt
twonr.tal1 = temp;   // till structen utan använd temp.
printf("Ge ett tal mellan 0 och 15 : ");
scanf("%u", &temp);
twonr.tal2=temp;
summa = twonr.tal1 + twonr.tal2;
printf("Summa = %u", summa);
```

(Ex) Variabeln tal innehåller två tal i vardera 16 bitar enligt:

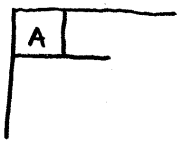


Plocka ut talen ur minnet och skriv ut de enskilda talen samt deras summa.

```
struct byte_4
{
    unsigned int tal1 : 16;
    Unsigned int tal2 : 16;
} *b4sp = (struct byte_4 *) &tal;

unsigned int summa;

printf("Tal1 = %u\n", b4sp->tal1);
printf("Tal2 = %u\n", b4sp->tal2);
Summa = b4sp->tal1 + b4sp->tal2;
printf("Summan = %u\n", summa);
```

Hårdvarunära programmering

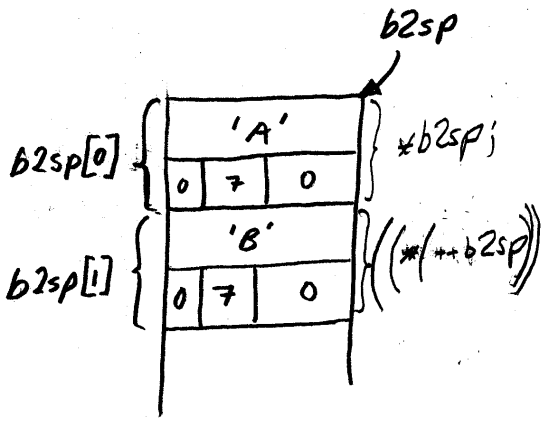
(Ex) Antag att skärmen till en dator avbildas i minnet med början på adressen 0xb8000000 enligt:



Skärm	Minesadress	Innehåll
	0xb8000000	'A' (Teckenfärg 0–15, svart-vitt)
	— " — 01	0111 0000
		Blink (↑) Bakgrundsfärg 0–7 svart-vitt (↑)
	0xb8000002	'B'
	— " — 03	0111 0000

```
volatile // säkrar att variabeln ej optimeras till register
struct byte-2
{   unsigned int tecken : 8;
    unsigned int teckcol : 4;
    unsigned int backcol : 3;
    unsigned int blink : 1;
} *b2sp = (struct byte-2 *) 0xb8000000;
```

```
b2sp -> tecken = 'A';
b2sp -> teckcol = 0;
b2sp -> backcol = 7;
b2sp -> blink = 0;
```




```
b2sp[1].tecken = 'B';
b2sp[1].teckcol = 0;
b2sp[1].backcol = 7;
b2sp[1].blink = 0;
```

(Ex) Fyll hela skärmen 80*25 tecken med blinkade röda stjärnn.

```
for(i=0; i < 25*80; i++)
{
    b2sp[i].tecken = '*';
    b2sp[i].teckcol = 5;
    b2sp[i].backcol = 0;
    b2sp[i].blink = 1;
```

Frågeoperatorn

(Ex) Beräkna största och minsta värdet av två heltal.

a)
```
if (a>b)
{
    max=a;
}
else
{
    max=b;
}
```

b)
```
switch (a>b)
{
    case 1: max=a;
            break;
    case 0: max=b;
            break;
}
```

c)
```
max=(a>b)? a:b; //Frågeoperatorn
```