EXEMPEL 1

EXEMPEL: Vi har deklarationerna:

```
int i,j;
int veci[80];
int vm[10][5];
```

Visa kodsekvenser som evaluerar följande uttryck till register R0.

a) `veci[j];`

b) `vm[i][j]`

Vi löser på tavlan...

a)

```
@ minnesadress: =veci + j*sizeof(int)
    LDR     R0,j          @ R0 ← j
    LSL     R0,R0,#2      @ R0 ← j*sizeof(int)=(j*4)
    LDR     R1,=veci      @ R1 ← =veci
    LDR     R0,[R1,R0]    @ R0 ← M(=veci+(4*j)= veci[j]
```

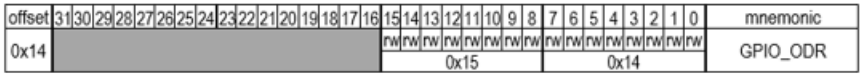
b)

```
@ minnesadress: =vm +( (i*5) + j )* sizeof(int)
    LDR     R3,i
    LSL     R2,R3,#2      @ R2 ← i*4
    ADD     R3,R2,R3      @ R3 ← (i*4)+i = (i*5)
    LDR     R2,j          @ R2 ← j
    ADD     R3,R3,R2      @ R3 ← (i*5)+j
    LSL     R3,R3,#2      @ R3 ← ((i*5)+j)*sizeof(int)
    LDR     R2,=vm
    LDR     R0,[R3,R2]    @ R0 ← M(int) ( ((i*5)+j)*4 )
```

EXEMPEL 2

Exempel:
Visa, såväl i C som i assembler, hur de fyra minst signifikanta bitarna hos en läs- och skrivbar port på address 0x40021014 sätts till 1, medan övriga bitar lämnas opåverkade.

Vi löser på tavlan




```
#define GPIO_ODR ((volatile unsigned short *)0x40021014)
    *GPIO_ODR |= 0xF;
```

```
LDR     R3,=0x40021014 @ R3 = 0x40021014
LDRH    R2,[R3]        @ R2 = (short) M(0x40021014)
MOV     R3,#0xF        @ R3 = 0xF
ORR     R2,R2,R3       @ R2 = (short) M(0x40021014) | 0xF
STRH    R2,[R3]        @ M(0x40021014) = (short) M(0x40021014) | 0xF
```

Pekararitmetik

Exempel:
I standard-C biblioteket finns funktionen **`strcpy`**, för att kopiera en ASCII-sträng i minnet. Strängslut markeras med tecknet `'\0'`, dvs. 0 och detta tecken ska vara det sista som kopieras. Funktionen returnerar `dest` och kan implementeras på följande sätt, visa hur **`strcpy`** kan kodas i assemblerspråk.

```
char *strcpy( char *dest, char * src )
{
  char *rval = dest;
  do{
    *dest++ = *src++;
  } while( *(src-1) );
  return rval;
}
```

Vi löser på tavlan...

```
strcpy:
@ Registeranvändning:
@   R0= &dest
@   R1= &src
@   R2= temporärregister
@   R3= lokal variabel rval
    MOV     R3,R0
.L0:
    LDRB    R2,[R1]
    STRB    R2,[R0]
    ADD     R1,R1,#1
    ADD     R0,R0,#1
    CMP     R2,#0
    BNE     .L0
    MOV     R0,R3
    BX      LR
```