

a)

@ minnesadress: =veci + j*sizeof(int)

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

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...

EXEMPEL 2

Exempel:

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

Vi löser på tavlan

offset	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	mnemonic									
0x14																rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	GPIO_ODR
																0x15							0x14																			

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

```
LDR    R3,=0x40021014 @ R3 = 0x40021014
LDRH   R2,[R3]         @ R2 = (short) M(0x40021014)
MOV     R1,#0xF         @ R3 = 0xF
ORR     R2,R2,R1        @ R2 = (short) M(0x40021014) | 0xF
STRH    R2,[R3]         @ M(0x40021014) = (short) M(0x40021014) | 0xF
```