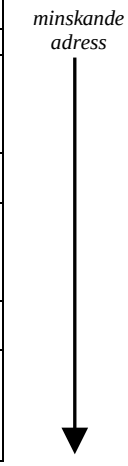
Lösningsförslag

Uppgift 1a:

Beskrivning av aktiveringspost

minnesanvändning	*adress*
ceasar	15,SP
bertil	14,SP
adam	10,SP
PC (vid JSR)	
a	4,SP
b	2,SP
c	0,SP




```
;  void func(long adam, unsigned char bertil, struct one * ceasar )
_func:
;  {
    LEAS   -8,SP
;   long               a = adam;
    LDY    10,SP
    LDD    12,SP
    STY    4,SP
    STD    6,SP
;   unsigned int   b = bertil;
    LDAB   14,SP
    CLRA
    STD    2,SP
;   struct one     *c = ceasar;
    LDD    15,SP
    STD    0,SP
;   /* Övrig kod i funktionen är bortklippt eftersom vi bara
;       betraktar anropskonventionerna. */
;  }
    LEAS   8,SP
    RTS
```

Uppgift 1b:

```
_c:     RMB 2   ;   struct one * c;
_a:     RMB 4   ;   long           a;
_b:     RMB 1   ;   char           b;
```

Funktionsanrop:

```
    LDD     _c
    PSHD
    LDAB    _b
    PSHB
    LDD     _a
    LDY     _a+2
    PSHD
    PSHY
    JSR     _func
    LEAS    7,SP
```

Uppgift 2:

```
_bitcount:
;     4 | {
;   Registerallokering:
;       reg B: value
;       reg A: count
;     5 |    unsigned char count;
;     6 |
;     7 |    for ( count=0; value != 0; value >>= 1)
    CLRA            ; count=0
bitcount_2:
    TSTB            ; value != 0
    BNE     bitcount_4
    BRA     bitcount_5

bitcount_3:
    LSRB            ; value >>= 1
    BRA     bitcount_2

bitcount_4:
;     8 |    {
;     9 |       if (value & 01)
    BITB    #1
    BEQ     bitcount_3
;    10 |          count++;
    INC A
;    11 |    }
    BRA     bitcount_3

bitcount_5:
;    12 |
;    13 |    return count;
;    14 | }
    TFR     A,B
    RTS
```

Uppgift 3:

```
typedef unsigned char * port8ptr;
#define OUT *((port8ptr) 0x400)
#define IN  *((port8ptr) 0x600)

void    EvenIndicator( void )
{
    unsigned char count, portvalue;

    portvalue = IN;
    count = 0;
    while( portvalue )
    {
        if( portvalue & 1 )
            count++;
        portvalue = portvalue >> 1;
    }
    if( count & 1 )
        OUT = 1;
    else
        OUT = 0x80;
```

Uppgift 4:

```
#define IRQ_STATUS ((unsigned char *) 0x800)
 void atIRQ( void )
 {
    if( (*IRQ_STATUS & 0x80 ) == 0 )
        return;
    if( *IRQ_STATUS & 1 )
    {   /* inpassering */
        visitors++;
        *IRQ_STATUS |= 4;
    }
    if( *IRQ_STATUS & 2 )
    {   /* utpassering */
        visitors--;
        *IRQ_STATUS |= 8;
    }
}
```

Uppgift 5:

```
long int strtol_hex (const char *str, const char **endptr) {
    const char *s = str;
    int c;
    int sum = 0, neg = 0;
    do {
        c = *s++;
    } while (c == ' ' || c == '\t' || c == '\n');
    if (c == '-') {
        neg = 1;
        c = *s++;
    }
    else if (c == '+')
        c = *s++;
    if (c == '0' && (*s == 'x' || *s == 'X')) {
         s++;
        c = *s++;
    }
    while (1) {
      if (c >= '0' && c <= '9')
        c = c - '0';
      else if (c >= 'a' && c <= 'f')
        c = c - 'a' + 10;
      else if (c >= 'A' && c <= 'F')
        c = c - 'A' + 10;
      else
        break;
      sum = sum * 16 + c;
      c = *s++;
    }
    if (neg)
      sum = -sum;
    if (endptr)
      *endptr = s-1;
    return sum;
}
```

Uppgift 6:

Filen med avbrottsrutinen (assembler):

```
        segment text
        export  _clocktrap
        import  _clock_inter

_clocktrap:
        JSR  _clock_inter
       RTI
```

Filen `clockports.h`:

```
typedef unsigned char port;        // port
typedef port *portptr;             // pekare till en port
typedef void (*vec) (void);        // avbrottsvektor
typedef vec *vecptr;               // pekare till en avbrottsvektor
#define setbits(r, mask)    (r) = (r) | (mask)
#define CRG_VEC_ADR      0xFFF0    // Adress till avbrottsvektor
#define CRG_VEC          *((vecptr) CRG_VEC_ADR)
#define CRGFLG_ADR      0x37       // För att kvittera avbrott
#define CRGFLG          *((portptr) CRGFLG_ADR)
#define rtif_bit        0x80
#define CRGINT_ADR      0x38       // För att aktivera avbrott
#define CRGINT          *((portptr) CRGINT_ADR)
#define rtie_bit        0x80
#define RTICTL_ADR      0x3B       // För att ange avbrottsintervall
#define RTICTL          *((portptr) RTICTL_ADR)
#define TIME_INTERVAL   10         // Antal ms mellan avbrott
#define TIME_CODE       0x49       // Ger ungefär ovanstående
```

Filen `scheduler.c`:

```
#include "scheduler.h"
#include "clockports.h"
#define N 100

struct tab_struct {
  function f;
  unsigned long int interval;
  unsigned long int next_time;
} tab[N];

static int num = 0;                        // antalet registrerade funktioner
```

```
static int busy = 0;                      // anger om någon funktion exekveras
static unsigned long int current_time = 0;  // tick, räknas upp vid varje avbrott

void init_scheduler(void) {
  current_time = 0;             // Initiera räknaren
  extern void clocktrap();      // Avbrottsrutinen (assembler)
  CRG_VEC = clocktrap;          // Initiera avbrottsvektor
  setbits(CRGINT, rtie_bit);    // Aktivera timer-avbrott
  RTICTL = TIME_CODE;           // Sätt verklig avbrottsfrekvens
}

void clock_inter(void) {        // anropas vid avbrott
  current_time++;
  setbits(CRGFLG, rtif_bit);
  if (!busy && num > 0) {
    busy = 1;
    // sök den funktion som ska anropas först
    int first = 0;
    for (int i=1; i<num; i++)
      if (tab[i].next_time < tab[first].next_time)
        first = i;
    if (tab[first].next_time <= current_time) {  // är det dags?
      tab[first].f();
      tab[first].next_time += tab[first].interval;
    }
    busy = 0;
  }
}

int schedule(function f, unsigned long int interval) {
  if (num < N) {
    tab[num].f = f;
    tab[num].interval = interval / TIME_INTERVAL;  // skala om från ms -> tick
    tab[num].next_time = current_time + tab[num].interval;
    num++;
    return 1;
  }
  else return 0;
}
```