Departament de Disseny i Programació de Sistemes Electrònics
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Final de PBN

18 de juny de 2014 — Enginyeria de Sistemes TIC — 120 MINUTS

COGNOMS:

NOM: GRUP de LAB:

Exercici 1 [1 punt]. Considereu el següent fragment de codi C:

```
uint8_t compute(int8_t a) {
  int8_t d = 4;

  d += a << 1;
  return d--;
}
```

El cos de la funció conté una operació que no aporta res. Indica quina i per què.

Exercici 2 [1 punt]. Considereu el següent fragment de codi C:

```
typedef struct {uint8_t l; int8_t t[20];} t1;
t1 a,*b;
uint8_t x,y;
```

Quina de les següents expressions és correcta?

- ○ x = *(b->t)
- ○ t1.l = 6
- ○ b.t[x] = y
- ○ x < a.t

Exercici 3 [1 punt]. Considereu el següent fragment de codi C corresponent a un programa per l'AVR:

```
uint8_t a,b;

a = PORTA;
b = (a * 4) & 0xf0;
```

Quina de les següents sentències és certa?

- ◯ El codi és erroni atès que el producte per 4 pot provocar un overflow.
- ◯ A la darrera assignació hi haurà una promoció de tipus que converteix a a tipus **int**.
- ◯ L'operador & és erroni i cadria escriure'l com &&.
- ◯ El fragment és erroni atès que multiplica un uint8_t per un **int**, cosa que no es pot fer.

Exercici 4 [1 punt]. Considereu un mòdul C per l'AVR que conté, entre altres, una funció main com la que veieu a continuació. En compilar el mòdul la funció main es tradueix a l'assemblador com s'indica literalment al costat:

```
int main (void) {
  uint8_t counter;
  DDRB = 0xFF;
/* port B en mode output */

  for (;;) {
    PORTB = 0xFF;
    retard(5);
    PORTB = 0x00;
    retard(100);
  }
  return 1;
}
```

```
main:
    ldi  r24,lo8(-1)
    out  0x4,r24
    ldi  r28,lo8(-1)
.L2:
    out  0x5,r28
    ldi  r24,lo8(5)
    call  retard
    out  0x5,__zero_reg__
    ldi  r24,lo8(100)
    call  retard
    rjmp .L2
```

Determineu i justifiqueu: (a) si hi ha hagut optimització; (b) si retard s'havia implementat com una funció o com una macro del preprocessador.

Exercici 5 [1 punt]. Considereu mòdul `C` per l'AVR i el resultat de compilar-lo amb la flag −Os.

```
#include <inttypes.h>

int8_t f1(int8_t a) {
  return (a << 2) + 1;
}

int8_t f2(int8_t a) {
  int8_t x = 3;

  x += f1(x);
  return --x;
}
```

```
.file   "modul.c"
__SP_H__ = 0x3e
__SP_L__ = 0x3d
__SREG__ = 0x3f
__tmp_reg__ = 0
__zero_reg__ = 1
    .text
.global f1
    .type   f1, @function
f1:
    lsl  r24
    lsl  r24
    subi r24,lo8(-(1))
    ret
    .size   f1, .-f1
.global f2
    .type   f2, @function
f2:
    ldi  r24,lo8(15)
    ret
    .size   f2, .-f2
    .ident  "GCC: (GNU) 4.7.2"
```

Identifiqueu les optimitzacions que s'han dut a terme i comenteu la seva oportunitat. En cas que la funció f1 hagués estat qualificada amb **static**, s'haguéssin pogut aplicar noves optimitzacions? quines?

Exercici 6 [1 punt]. En el cas del mòdul del problema anterior, quantes entrades tindrà la taula de reubicacions del mòdul? per què?

Exercici 7 [2 punts]. Un executable està format pels mòduls a.o i b.o i c.o que s'enllacen en aquest ordre precís. La taula de reubicacions i el codi màquina del mòdul b.o són:

```
RELOCATION RECORDS FOR [.text]:
OFFSET   TYPE              VALUE
0000000a R_AVR_CALL        fc
```

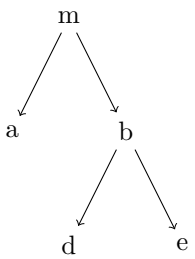
```
Disassembly of section .text:
   0: 88 0f
   2: 88 0f
   4: 8f 5f
   6: 08 95
   8: 80 e1
   a: 0e 94 00 00
  10: 80 7f
  12: 81 50
  14: 08 95
```

D'altra banda, la taula de símbols del mòdul c.o és:

```
SYMBOL TABLE:
00000000 l    df *ABS* 00000000 c.c
00000000 l    d  .text 00000000 .text
00000000 l    d  .data 00000000 .data
00000000 l    d  .bss 00000000 .bss
0000003e l       *ABS* 00000000 __SP_H__
0000003d l       *ABS* 00000000 __SP_L__
0000003f l       *ABS* 00000000 __SREG__
00000000 l       *ABS* 00000000 __tmp_reg__
00000001 l       *ABS* 00000000 __zero_reg__
00000000 l    d  .comment 00000000 .comment
00000000 g     F .text 00000004 fc
```

Assumint que en el moment d'enllaçar els objectes el mòdul c.o s'ubica a l'adreça 0x0096, escriviu el codi màquina del mòdul b.o una vegada reubicat.

Exercici 8 [1 punt]. Com sabeu el diagrama de mòduls d'un projecte i les dependències del `Makefile` del mateix projecte no són el mateix però tenen una relació. Assumint que un projecte respòn al següent diagrama de mòduls, i que la estratègia d'organització del codi és l'habitual, quines serien les dependències del `Makefile` corresponent?



Exercici 9 [2 punts]. Considereu el següent fragment de codi d'un programa per l'AVR:

```
...

static volatile bool run = false;

ISR(TIMER0_OVF_vect) {
  if (!run) run = true;
}
```

```
int main(void) {
  ...
  while (!run) ;
  run = false;
  return x;
}
```

Explica quin paper juga l'atribut **volatile** i justifica, si és el cas, la seva necessitat.

Exercici 10 [2 punts]. Situeu-vos en el context de les pràctiques. Com sabeu finalment s'acaben implementant dos dispositius que es comuniquen entre ells usant codi Morse en la banda audible. Argumenteu quins són els factors software més importants que limiten la velocitat de transmissió entre aquests dos dispositius.