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

Final de PBN

20 de juny de 2016 — Enginyeria de Sistemes TIC — 120 MINUTS

COGNOMS:

NOM: GRUP de LAB:

Exercici 1 [2 punts]. El següent fragment de codi correspon íntegrament al resultat de compilar un fitxer font `C`.

```
.file   "a.c"
    __SP_H__ = 0x3e
    __SP_L__ = 0x3d
    __SREG__ = 0x3f
    __tmp_reg__ = 0
    __zero_reg__ = 1
    .text
    .global fuzzyfunc
    .type   fuzzyfunc, @function
fuzzyfunc:
    /* prologue: function */
    /* frame size = 0 */
    /* stack size = 0 */
    .L__stack_usage = 0
    ldi r25,0
.L2:
    tst r24
    breq .L5
    subi r25,lo8(-(1))
    sub r24,r22
    rjmp .L2
.L5:
    mov r24,r25
    ret
    .size   fuzzyfunc, .-fuzzyfunc
    .ident  "GCC: (GNU) 4.8.1"
```

Contesteu les següents preguntes:

1. El fitxer que s'ha compilat era un programa?

2. Per què es fa un `tst r24` quan no s'ha inicialitzat el registre amb cap valor?

3. El cos principal d'aquest codi és la traducció d'una sentència de control de flux en el font original. A quina sentència podria correspondre aquesta traducció?

4. En el font, la funció fuzzyfunc() retornava **void** ?

Exercici 2 [1 punt]. El *loop unrolling* és una estratègia d'optimització usada pels compiladors que consisteix a traduir una iteració «desplegant-la», o sia repetint les vegades que sigui necessari el cos de la iteració. Estudieu la següent funció i determineu si és aplicable o no aquesta tècnica. Justifiqueu la resposta.

```
int to_unroll_or_not_to_unroll(int *const t, int i) {
  int k;

  for(k=0; k<*t; k++)
     i *= 2;

  return i;
}
```

Exercici 3 [1 punt]. Aquest exercici fa referència als concepte de *linkage* que es defineix a l'estàndart de C.

Considereu els dos fitxers font següents que formen part d'un mateix programa.

```
/* Modul A */
#include <stdlib.h>

static int a;
extern int a;
...
```

```
/* Modul B */
#include <stdlib.h>

int a;
...
```

Determineu (1) si l'identificador a en els mòduls A i B fa referència a objectes diferents o al mateix objecte i (2) si la declaració d'a en el mòdul B és una definició o no i perquè.

Exercici 4 [2 punts].

Assumiu que un projecte de desenvolupament està estructurat amb els mòduls A, B, C i P, essent P el mòdul que conté el programa principal. Les dependències entre mòduls són les següents: $P \xrightarrow{usa} B$, $P \xrightarrow{usa} A$, $A \xrightarrow{usa} C$.

Si el projecte s'implementa amb el llenguatge C i la forma d'implementar el concepte de mòdul és la estàndart de l'assignatura, quin serà el graf de dependències entre fitxers? I la descripció de les dependències en un `Makefile` assumint les regles per defecte convencionals?

Exercici 5 [2 punts]. Una làmpada d'un far s'activa/desactiva amb les funcions

```
void lamp_on(void);
void lamp_off(void);
```

Dissenyeu i implementeu una aplicació, obligatoriament basada en el mòdul `timer` de la pràctica, que governi el far i li doni una seqüència cíclica de 1.2 s encès, 2 s apagat, 2 s encès i 1 s apagat.

Exercici 6 [2 punts]. Assumiu que es compila usant l'ordre

```
avr-gcc -c -Os -mmcu=atmega328p -DF_CPU=16000000UL b.c
```

el modul següent:

```
static int a;

int f1(int z) {
  return (z > 0)?(z+1):(z-1);
}
```

Després de compilar, la taula de símbols del mòdul objecte és la que segueix:

```
b.o:     file format elf32-avr

SYMBOL TABLE:
00000000 l    df *ABS* 00000000 b.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 0000000e f1
```

Hi hauria de sortir la variable a? Per quina raó hi surt o no hi surt?