Cognoms: . . . . . Nom: . . . . . .

1er Control Arquitectura de Computadors **Curs 2013-2014 Q1**

Problema 1. (4 punts)

Volem produir un sistema empotrat per correr una única aplicació a partir d'un processador dissenyat per a aplicacions de "IoT" (Internet of Things) que pot operar correctament a freqüències de rellotge entre 100 i 600MHz sense modificar el voltatge. El nostre generador de rellotge només permet generar freqüències que siguin múltiples de 100Mz. El bucle infinit del programa principal del nostre sistema executa 7.654.321 instruccions dinàmiques per iteració i té un CPI de "6,71 + Freq_en_MHz/300".

a) **Calcula** a quina freqüència vàlida mínima ha d'operar el nostre sistema empotrat si volem que una iteració del bucle del programa principal tardi menys de 0,2 segons

El processador opera a 1,2V, el consum dinàmic (potencia) es inferior a 0.3mW per MHz i la corrent de fuita es despreciable.

b) **Calcula** quina serà la potència màxima del nostre sistema

La carrega d'una pila o bateria es mesura en Ampers*hora (Ah), aquesta mètrica indica la quantitat de corrent que pot entregar de forma contínua normalitzat a una hora (abans de quedar descarregada totalment).

c) **Calcula** quant de temps de forma continuada, com a mínim, podrà operar el nostre sistema empotrat si està alimentat per una pila de 1,2V amb una carrega de 1,8Ah

Volem reduir el temps d'execució d'una iteració del bucle principal del programa a la meitat sense modificar la freqüència de treball del nostre processador i, per tant, decidim construir un multiprocessador i paral.lelitzar el codi del bucle principal.

d) Quin es el número mínim de processadors com el dels apartats anteriors que necessitem en el nostre multiprocessador per a resoldre el problema si, en el millor cas, la fracció del temps original que es pot executar en paral.lel es del 81%?

Cognoms: . . . . . Nom: . . . . . .

1er Control Arquitectura de Computadors **Curs 2013-2014 Q1**

Problema 2. (3 puntos)

Dado el siguiente código escrito en C, que compilamos para un sistema linux de 32 bits:

```
typedef struct {
  char a;
  int b;
  short c;
  char d;
  int e[5];
} s1;
```

```
typedef struct {
  char c;
  s1 f[100];
  int d;
  } s2;
```

a) **Dibuja** como quedarían almacenadas en memoria las estructuras **s1** y **s2**, indicando claramente los deplazamientos respecto al inicio, el tamaño de todos los campos y el tamaño de los structs.

b) **Escribe** la expresión aritmética que permite calcular la dirección del elemento **x.f[y.e[i]].d**, siendo **x** una variable de tipo **s2** e **y** una variable de tipo **s1**:

c) **Traduce** la siguiente sentencia a ensamblador del x86, suponiendo que **py** es un puntero a un struct **s1**. El puntero **py** se encuentra almacenado en **8(%ebp)** y la variable **i** en **-4(%ebp)**. Se valorará la correcta utilización de los modos de direccionamiento y el uso del mínimo número de instrucciones.

```
py->e[i]=0;
```

d) **Traduce** la siguiente sentencia a ensamblador del x86, suponiendo que **px** es un puntero a un struct **s2**. El puntero **px** se encuentra almacenado en **8(%ebp)**. Se valorará la correcta utilización de los modos de direccionamiento y el uso del mínimo número de instrucciones.

```
px->f[10].d = 'a';
```

e) **Define en C** una estructura equivalente a **s1**, reordenando sus campos de forma que se optimice el espacio ocupado en memoria. Indica cuántos bytes de memoria se ahorran al almacenar **s2**. :;

Cognoms: . . . . . Nom: . . . . . .

1er Control Arquitectura de Computadors **Curs 2013-2014 Q1**

Problema 3. (3 puntos)

a) **Explica** en qué consiste el enlace dinámico de una subrutina, por qué es necesario y cómo se hace (escribe las instrucciones que hacen el enlace dinámico)..

Dadas las siguientes definiciones de variables globales en un programa en C:

```
int Matriz[50][100],Vector[100],i,j,suma;
char MC[50][100],VC[100];
```

b) **Escribe** una secuencia de 2 INSTRUCCIONES que permita guardar en los 8 bits de menor peso del registro **%eax** el valor de **MC[i][j]**, sabiendo que **i** está en el registro **%edi** y **j** en el registro **%esi**. Indica claramente la fórmula que te permite calcular la dirección del elemento **MC[i][j]** a partir de la dirección simbólica **MC** (que es la del elemento **MC[0][0]**).

c) **Escribe** una secuencia de 4 INSTRUCCIONES que permita guardar en **Vector[j]** el valor de **Matriz[i][j]**, sabiendo que i está en el registro **%edi** y j en el registro **%esi**. Indica claramente la fórmula que te permite calcular la dirección de los elementos **Vector[j]** y **Matriz[i][j]** a partir de las direcciones simbólicas **Vector** y **Matriz** (que son las de los elementos **Vector[0]** y **Matriz[0][0]** respectivamente).

d) **Traduce** literalmente a ensamblador del x86 (sin optimizar las estructuras de control) el siguiente código escrito en C. Se valorará el correcto uso de los modos de direccionamiento.

```
for (i=0,suma=0;i<99;i++) {
  if (Vector[i]>Vector[i+1]) {
      suma=suma+Vector[i];
  }
  else suma=0;
}
```