Universitat de Girona - Escola Politècnica Superior

COMPUTADORS EXAMEN PARCIAL Curs 2003-04 - 15 de Juny de 2004.

Examen Problemes: Durada 2h 30 min. **Si us plau, feu els exercicis en fulls separats.**

Les notes es publicaran el dia 22 de Juny als taulells de PII i a la web de l'assignatura. La revisió de l'examen es realitzarà el 24 de Juny de 11h a 13h a l'aula II-09.

1- 'Teclat'.

A) Descriu les modificacions necessàries en el maquinari (la placa del laboratori) per tal de controlar un teclat matricial de cinquanta-vuit tecles. Aquest teclat serà alfanumèric (tipus 'QWERTY') amb unes tecles de 'funcions especials' (Control, '*Escape*', Majúscula, Tabulador, Retorn, fletxes de desplaçament...). La disposició d'aquestes tecles encara està 'per decidir' però sabem que seran polsadors 'normalment oberts' de dos contactes muntats sobre una sola placa de circuit imprès (el microcontrolador anirà muntat en una altra placa).

B) Descriu l'algorisme i les estructures de dades necessàries per tal de controlar la pulsació simultània de dues tecles, el *'rollover'*, (Cas de 'CTRL+X', 'Majúscula+a'...). Afegiu totes les consideracions (o restriccions) que creieu convenients.

C) Implementeu (en C) la funció *'int escombra_teclat(void);'* que retorna les dues tecles premudes (el 'pitjor' cas). Cadascuna d'aquestes tecles serà un valor de 8 bits (i per això retorna un *'int'*, que són 16 bits) corresponent al codi ASCII de la tecla premuda (en cas que aquesta tingui 'representació'). Es retornarà el valor de les tecles 'alfanumèriques' per la part baixa i les 'especials' per la part alta. En cas de pulsacions 'simples' (només una tecla premuda) el byte alt del valor de retorn serà zero. No cal que filtreu els rebots, aquesta tasca s'efectua a un nivell superior. Podeu comptar que ja està implementada la funció *'char escombra_fila(char columna);'* que retorna 'uns' en les tecles premudes que intersecten les files amb la columna passada com a paràmetre. Cal que retorneu '0xff' en cas d'error (més de dues tecles pitjades).

2- 'Displai'.

Per al producte final es necessita un visualitzador tipus LCD (semblant al que heu fet servir). La interfície d'aquest i la seva programació és idèntica al del laboratori. Físicament és un visualitzador de quatre línies de quaranta caràcters. La organització 'lògica' d'aquest és encara més senzilla: el programador veu un *'buffer'* de cent seixanta bytes (quatre per quaranta) en el que la primera línia correspon a les posicions 'de la zero a la trenta-nou', la segona línia 'de la quaranta a la setanta-nou', i així successivament. En aquest visualitzador no existeix el concepte de 'finestra que mostra una porció d'un búffer'.

A) Implementeu (en C) la funció *'void posicionat(char x, char y);'* que situa el cursor a la posició indicada com a paràmetres (el nom de la funció correspon al imperatiu *posiciona't*). La posició '(0,0)' correspon al primer caràcter de la primera línia i s'escriurà d'esquerra a dreta i de dalt a baix.

B) Implementeu (en C) la funció *'void escriu_cadena(char *cadena);'* que escriu una cadena de text a partir de la posició actual (fixada prèviament amb la funció anterior).

C) Implementeu (en C) la funció *'void scroll_displai(void);'* que efectua un *'scroll vertical'* d'una línia. Això és: el text que hi ha a la segona línia es copia sobre la primera, el que hi ha a la tercera sobre la segona i així successivament fins a la última, la última línia ha de quedar en blanc. D'aquesta manera, aparentment, tot el text es desplaça una línia amunt.

Nota: En tots els apartats especifiqueu de forma ben clara les necessitats que es tenen i la manera de resoldre-les. Podeu suposar implementades les funcions de més baix nivell , tan sols indiqueu la seva funcionalitat.

3- 'Sèrie seriós'.

Volem controlar períodes de temps molt precisos amb un PC (de l'ordre de mili-segons), i per això no ens serveix l'anomenada 'interrupció periòdica' (que es produeix 18,2 vegades per segon i més val no tocar). La manera que se'ns acut és muntar un oscil·lador i connectar-lo a una de les línies del port sèrie; concretament al *'RI'* (*'Ring Indicator'*) senyal amb la que un módem indica al computador que la línia rep una trucada ('tuuuut, tuuuut, tuuuut....). Suposeu que al pin *'RI'* hi ha connectat un senyal digital quadrat de freqüència cinc kilocicles.

A) Implementeu (en C) la funció *'void inicialitza_UART(void);'* de manera que s'inicialitzi el port sèrie per a una configuració de 38.400 bauds, 8 bits de dades, 1 bit de stop i sense paritat (38400,N,8,1) en la que només es produirà interrupció quan es rep un caràcter i quan hi ha una 'trucada' (el nostre 'truc' no tindrà res a veure amb la línia telefònica). Feu-ho per 'COM2', situat a l'adreça 0x2f8.

B) Implementeu (en C) la rutina de servei a la interrupció del port de manera que si s'ha rebut un caràcter s'emmagatzemi en un *'buffer'* (encua(caracter)) i en cas de 'trucada periòdica' decrementi una variable (per ex. 'temps') segons l'estat d'una altra de control (per ex. 'marxa'). Si aquesta variable arriba a zero es senyalitza amb un 'flag' (per ex. 'timeout'). D'aquesta manera, des del programa principal podrem 'temporitzar' de forma precisa amb el següent codi:

```
temps=25; timeout=fals; marxa=cert; /* Poso en marxa una temporització */
while (!timeout) { /* no faig res */};
marxa=fals; /* han passat 25 unitats de temps */
```

Nota: indiqueu els tipus de les variables, la seva ubicació i tot el referent a les unitats de temps per tal de tenir una resolució de mili-segons.

4- 'Cau-cache'.

El 80486 té una *'cache'* (o cau) dins la mateixa pastilla del processador. Aquesta és de vuit kilobytes segons una organització associativa de quatre vies i longitud de bloc de quatre paraules de trenta-dos bits. El controlador s'estructura en cent vint-i-vuit conjunts. Hi ha un bit de 'línia vàlida' i tres bits (B0, B1 i B2) de 'LRU' (*'Least Recently Used'*) per línia (o *'bank'*). En cas de errada de *'cache'*, el 486 llegeix setze bytes de memòria principal en una ratxa (*'burst'* o *'ráfaga'*) de lectura de memòria principal a través del bus.

A) Dibuixeu un diagrama de la *'cache'* i indiqueu com s'interpreten els diferents camps d'adreces.

L'algorisme de substitució s'anomena *'pseudo-LRU'*. Associat a cadascun dels cent vint-i-vuit conjunts de quatre línies (o *'bank';* L0, L1, L2 i L3) hi ha tres bits (B0, B1 i B2). L'algorisme de substitució procedeix de la següent forma: quan s'ha de substituir una línia, primerament es determina si l'ús més recent ha estat de L0 i L1 o bé de L2 i L3. Llavors es determina quin element de la parella ha estat usat menys recentment i es substitueix.

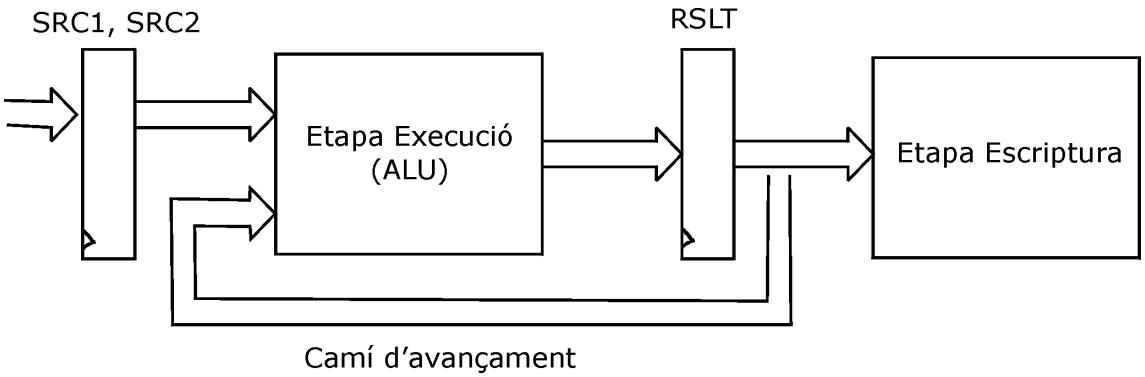
B) Especifiqueu com varien els bits B0, B1 i B2 i com s'utilitzen en l'algorisme de substitució. Demostreu que aquest algorisme s'aproxima a un LRU veritable. Demostreu que un algorisme LRU veritable requereix sis bits per conjunt.

5- 'Escales, súpers i execucions'.

Considereu el següent codi on la sintaxi consta d'un codi d'operació seguit d'un registre destí i dos fonts (o origen).

```
0     ADD   r3, r1,r2
1     LOAD  r6, [r3]
2     AND   r7, r5, r3
3     ADD   r1, r6, r7
4     SRL   r7, r0, 8
5     OR    r2, r4, r7
6     SUB   r5, r3, r4
7     ADD   r0, r1, 10
8     LOAD  r6, [r5]
9     SUB   r2, r1, r6
10    AND   r3, r7, 15
```

Suposeu una execució de quatre etapes: 'fetch' (cerca d'inst.), descodificació/emissió, execució i escriptura de resultats. Totes les etapes, excepte la d'execució, necessiten un cicle de rellotge. En les instruccions aritmètico-lògiques l'execució s'efectua en un cicle mentre que les de càrrega en necessiten tres. Suposeu una execució escalar senzilla (una única unitat d'execució) que permet l'execució desordenada i amb avançament d'operand en l'etapa d'execució. Això és; existeix un camí d'avançament entre la sortida del registre que hi ha després de l'etapa d'execució ('RSLT') i l'entrada de la unitat d'execució. D'aquesta manera, davant un cas de dependència de dades, no cal 'esperar' que el valor present en el registre que hi ha entre les etapes d'execució i escriptura sigui actualitzat per obtenir el valor d'origen a través dels registres que hi ha abans de l'etapa d'execució ('SRCi'). Aquesta idea queda representada a la següent figura:



A) Feu un diagrama de temps de l'execució del codi. Podeu indicar, en una taula (o de la manera que més us convingui), el cicle de rellotge en el que cadascuna de les instruccions inicia cada fase. Comenteu tots aquells aspectes 'rellevants' per a la comprensió del diagrama.

B) Refeu la taula suposant que no es pot executar de forma desordenada.

C) Quina millora (en percentatge) aporta, en aquest cas, l'execució desordenada?