I PRÀCTIQUES D' ESTRUCTURA DE COMPUTADORS

Objectius

- Familiaritzar-se amb l'entorn de desenvolupament (Visual Studio).
- Aprendre els conceptes bàsics del llenguatge de baix nivell de l'arquitectura Intel x86.
- Treballar en C i llenguatge Assembly Intel x86.

Material

- Microsoft Visual Studio 2019 Community. Disponible a:
  https://visualstudio.microsoft.com/es/vs/
- Documentació del entorn de desenvolupament i la seva configuració.
- Esquelet del Projecte de la pràctica que inclou el programa en C i l'esquelet del programa assemblador (disponible al Campus Virtual).

☐ La pràctica es realitzarà en 6 sessions.

☐ Hi ha una prova de validació individual.

☐ La pràctica està dividida en 3 nivells: **bàsic**, **mig** i **avançat**. Quan un grup acabi un nivell, haurà de presentar-lo al professor de pràctiques durant la sessió. **No es poden saltar nivells** (p.e. presentar directament el nivell mig sense haver presentat el bàsic), **ni presentar més d'un nivell per sessió i el professor obrirà en el campus virtual perquè pugueu accedir al següent nivell.**

☐ La nota de laboratori depèn del nivell assolit i la qualitat/funcionalitat de la solució proposada. El nivell **bàsic** dóna dret a una nota fins **6**. El nivell **mig** fins a **7**, i **l'avançat** fins a **8**. En el Campus Virtual disposeu d'un document on s'explica amb més detall el mètode d'avaluació de l'assignatura.

☐ Com es farà el lliurament del codi?

- Un cop finalitzat cada nivell, es farà un lliurament del codi en el campus virtual, **només del codi font comentat pracX.asm** (on X=1 pel bàsic, 2 pel mig i 3 per l'avançat)

☐ Com es realitzarà el lliurament de la memòria?

- Es farà el lliurament de la memòria que ha d'incloure tot el treball desenvolupat a les pràctiques i el codi font prac.asm comentat. Al campus virtual teniu disponible la plantilla de Latex, que podeu carregar a *overleaf* (*https://es.overleaf.com/*), per a escriure la memòria.

Ambdós arxius seran lliurats a Campus Virtual en un arxiu comprimit amb el següent format de nom:TORN-NIU1-NIU2-NIU3-[NIVELL].[zip/rar], on TORN és el vostre torn.

Ex: 411-1133922-2123444-122344.zip grup 411 del dilluns a les 12:30.

La data límit per a fer el lliurament de la memòria és el **29 de desembre de 2021** a les 23:55.

Enunciat de la pràctica

Joc del "Busca Mines"

La pràctica consisteix en implementar un joc de busca mines. Tenim un tauler quadrat de 64 caselles (files de 1 a 8 i columnes de A a H), i cada casella conté, o bé res (0), o bé una mina (1). En total s'amaguen 9 mines en el tauler. El jugador ha d'anar descobrint caselles sense fer esclatar les mines amagades. El jugador port triar entre dos tipus de tirada, o bé obrir una casella, o bé marcar/desmarcar una casella indicant que sospita que amaga una mina.
En el cas que efectuem una tirada del tipus obrir una casella, i la casella descoberta no conté una mina, s'indicarà el nombre de mines que hi ha al voltant de la posició descoberta. En cas que la casella contingui una mina, aquesta explota i finalitza el joc (si per mala sort la primera casella descoberta es una mina el joc pot acabar a la primera tirada).
En el cas que efectuem una tirada tipus marcar "mina", la casella es marcarà amb una mina, es decrementarà el comptador de mines i el joc seguirà normalment. Però, si la posició ja havia estat assenyalada com a "mina" i es torna a marcar, s'interpreta que es vol desmarcar la casella, de manera que es desmarca la suposada mina i s'incrementa el comptador de marques. En tot moment s'informarà a l'usuari del nombre de marques restants.
El joc finalitza quan s'han descobert totes les caselles sense que cap hagi explotat cap i s'han marcat totes les mines.
La pràctica consta d'un programa en C, que us donem fet (main.cpp) i NO HEU DE MODIFICAR. Aquest programa en C, genera un menú en pantalla i permet accedir a les diferents opcions del menú. Les diferents opcions del menú s'han d'implementar amb les subrutines en assemblador que us indiquem.
A més a més, teniu un programa en assemblador (prac.asm) que conté algunes subrutines ja implementades per a gestionar l'entrada/sortida i l'estructura bàsica de les subrutines que heu d'implementar.

Treball a realitzar en el nivell bàsic (2 sessions)

En aquest primer nivell s'han d'implementar unes funcionalitats bàsiques, és a dir les 5 primeres opcions del menú. Aquests opcions són:

1**.- Show Cursor** - **Posicionar el cursor**: Posicionar el cursor a la pantalla, dins el tauler, en funció de les variables (*row*) fila (int) i (*col*) columna (char), a partir dels valors de les constants *RowScreenIni* i *ColScreenIni*. El cursor s'ha de posicionar a la posició indicada en el programa en C (fila 5, columna C), tingueu en consideració que la primera posició de la matriu és la (1, "A")). Haureu de completar la funció en ensamblador ***posCurScreenP1***.

2.- **Move**: En aquesta opció el cursor s'ha de moure una cel·la del caseller, utilitzant les següents tecles per indicar la direcció: "*i*" cap dalt, "*j*" cap a l'esquerra, "*k*" cap baix i "*l*" cap a la dreta.

En aquesta opció les funcions en ensamblador a desenvolupar són ***getMoveP1*** i ***moveCursorP1.*** Podeu incorporar també la lectura de les tecles "*m*" per a marcar una mina, " " per a descobrir una cel·la i "*s*" per sortir que haureu de fer servir en properes opcions. Recordeu comprovar els límits del taulell abans de fer un moviment i llegiu atentament el codi en C++ i les capçaleres del codi en ensamblador per saber les variables implicades.

3.- **Move continous**: Un cop incorporada la lectura de les tecles "*m*", " " i "*s*" , aquesta funció permet el moviment continu del cursor en el caseller finalitzant la seva execució en cas de que la tecla premuda no correspongui a un moviment. La funció a desenvolupar és ***movContinuoP1***.

4.- **Open:** En aquesta opció, un cop haguem situat el cursor sobre una cel·la s'ha d'obrir la casella si premem la tecla " " o marcar la casella amb una "***m***" si premem la tecla "*m*"; en prémer la tecla "*s*" sortirà sense fer cap acció. La funció a desenvolupar és ***openP1,*** tanmateix necessitaràs calcular la posició de la matriu ***mineField*** que correspon a la posició de la casella en la que es troba el cursor. Això ho hauràs d'implementar la funció ***calcIndexP1***. Per imprimir el caràcter corresponent a la posició indicada hauràs de cridar a la funció ***printch.***

5.- **Open continous:** En aquesta opció cal implementar l'obertura continua de caselles fins a prémer la tecla de sortida "s".

Implementació

Com ja hem dit, la pràctica consta d'un programa en C que us donem fet i **NO PODEU MODIFICAR**, i un programa en assemblador que conté algunes subrutines ja implementades (per posicionar el cursor, llegir caràcters de teclat i escriure caràcters per pantalla) i les capçaleres de les subrutines que heu d'implementar.

Subrutines que cal implementar en assemblador per al Nivell Bàsic:

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Posicionar el cursor a la pantalla, dins el tauler, en funció de
; les variables (row) fila (int) i (col) columna (char), a partir dels
; valors de les constants RowScreenIni i ColScreenIni.
; Primer cal restar 1 a row (fila) per a que quedi entre 0 i 7
; i convertir el char de la columna (A..H) a un número entre 0 i 7.
; Per calcular la posició del cursor a pantalla (rowScreen) i
; (colScreen) utilitzar aquestes fórmules:
; rowScreen=rowScreenIni+(row*2)
; colScreen=colScreenIni+(col*4)
; Per a posicionar el cursor cridar a la subrutina gotoxy.
;
; Variables utilitzades:
; row       : fila per a accedir a la matriu mineField/taulell
; col       : columna per a accedir a la matriu mineField/taulell
; rowScreen : fila on volem posicionar el cursor a la pantalla.
; colScreen : columna on volem posicionar el cursor a la pantalla.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;posCurScreenP1:
posCurScreenP1 proc
        push ebp
        mov  ebp, esp
        ;..... el teu codi aquí .......
        mov esp, ebp
        pop ebp
        ret
posCurScreenP1 endp
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Llegir un caràcter de teclat
; cridant a la subrutina getch
; Verificar que solament es pot introduir valors entre 'i' i 'l',
; o les tecles espai ' ', 'm' o 's' i deixar-lo a la variable carac2.
;
; Variables utilitzades:
; carac2 : Variable on s'emmagatzema el caràcter llegit
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;getMoveP1:
getMoveP1 proc
   push ebp
   mov  ebp, esp
   ;..... el teu codi aquí .......
   mov esp, ebp
   pop ebp
   ret
getMoveP1 endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Actualitzar les variables (rowCur) i (colCur) en funció de
; la tecla premuda que tenim a la variable (carac2)
; (i: amunt, j:esquerra, k:avall, l:dreta).
; Comprovar que no sortim del taulell, (rowCur) i (colCur) només poden
; prendre els valors [1..8] i [0..7]. Si al fer el moviment es surt
; del tauler, no fer el moviment.
; No posicionar el cursor a la pantalla, es fa a posCurScreenP1.
;
; Variables utilitzades:
; carac2 : caràcter llegit de teclat
;          'i': amunt, 'j':esquerra, 'k':avall, 'l':dreta
; rowCur : fila del cursor a la matriu mineField.
; colCur : columna del cursor a la matriu mineField.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;moveCursorP1: proc endp
moveCursorP1 proc
   push ebp
   mov  ebp, esp
   ;..... el teu codi aquí .......
   mov esp, ebp
   pop ebp
   ret
moveCursorP1 endp



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Subrutina que implementa el moviment continuo.
;
; Variables utilitzades:
;              carac2   : variable on s'emmagatzema el caràcter llegit
;              rowCur   : Fila del cursor a la matriu mineField
;              colCur   : Columna del cursor a la matriu mineField
;              row      : Fila per a accedir a la matriu mineField
;              col      : Columna per a accedir a la matriu mineField
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;movContinuoP1: proc endp
movContinuoP1 proc
       push ebp
       mov  ebp, esp
       ;..... el teu codi aquí .......
       mov esp, ebp
       pop ebp
       ret
movContinuoP1 endp
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Calcular l'índex per a accedir a les matrius en assemblador.
; mineField[row][col] en C, és [mineField+indexMat] en assemblador.
; on indexMat = row*8 + col (col convertir a número).
;
; Variables utilitzades:
; row       : fila per a accedir a la matriu mineField
; col       : columna per a accedir a la matriu mineField
; indexMat  : índex per a accedir a la matriu mineField
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;calcIndexP1: proc endp
calcIndexP1 proc
        push ebp
        mov  ebp, esp
        ;..... el teu codi aquí .......
        mov esp, ebp
        pop ebp
        ret
calcIndexP1 endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Obrim una casella de la matriu mineField
; En primer lloc calcular la posició de la matriu corresponent a la
; posició que ocupa el cursor a la pantalla, cridant a la
; subrutina calcIndexP1.
; En cas de que la casella no estigui oberta ni marcada mostrar:
;       - 'X' si hi ha una mina
;       - 'm' si volem marcar la casella
; En cas de que la casella estigui marcada mostrar:
;       - ' ' si volem desmarcar la casella
; Mostrarem el contingut de la casella criant a la subrutina printch. L'índex per
; a accedir a la matriu mineField, el calcularem cridant a la subrutina calcIndexP1.
; No es pot obrir una casella que ja tenim oberta o marcada.
;
; Variables utilitzades:
; row       : fila per a accedir a la matriu mineField
; rowCur    : fila actual del cursor a la matriu
; col       : columna per a accedir a la matriu mineField
; colCur    : columna actual del cursor a la matriu
; indexMat  : Índex per a accedir a la matriu mineField
; mineField : Matriu 8x8 on tenim les posicions de les mines.
; carac     : caràcter per a escriure a pantalla.
; taulell   : Matriu en la que anem indicant els valors de les nostres tirades
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;openP1: proc endp
openP1 proc
        push ebp
        mov  ebp, esp
        ;..... el teu codi aquí .......
        mov esp, ebp
        pop ebp
        ret
openP1 endp

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Subrutina que implementa l'obertura continua de caselles. S'ha d'utilitzar
; la tecla espai per a obrir una casella i la 's' per a sortir.
;
; Variables utilitzades:
; carac2    : Caràcter introduït per l'usuari
; rowCur    : Fila del cursor a la matriu mineField
; colCur    : Columna del cursor a la matriu mineField
; row       : Fila per a accedir a la matriu mineField
; col       : Columna per a accedir a la matriu mineField
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;openContinuousP1:proc endp
openContinuousP1 proc
        push ebp
        mov  ebp, esp
        ;..... el teu codi aquí .......
        mov esp, ebp
        pop ebp
        ret
openContinuousP1 endp
```