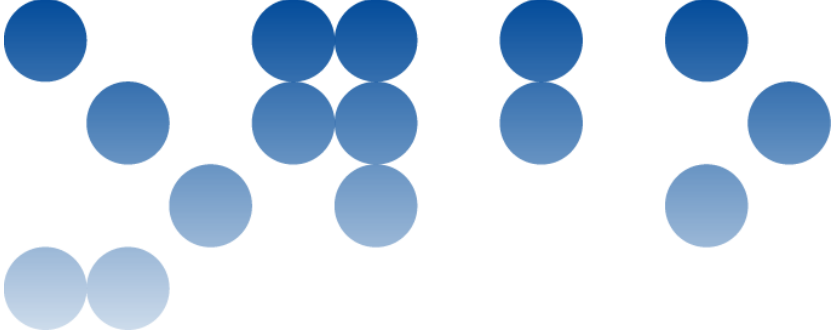

PRÀCTICA 1a part: Buscamines

Estructura de Computadors

Grau en Enginyeria Informàtica

set19-feb20

Estudis d'Informàtica, Multimèdia i Telecomunicació

EIMT.UOC.EDU

Universitat Oberta de Catalunya

Presentació

La pràctica que es descriu a continuació consisteix en la programació en llenguatge assemblador x86_64 d'un conjunt de subrutines, que s'han de poder cridar des d'un programa en C. L'objectiu és fer el joc del MasterMind.

La **PRÀCTICA** és una **activitat avaluable individual**, per tant no es poden fer comentaris molt amplis en el fòrum de l'assignatura. Es pot fer una consulta sobre un error que tingueu a l'hora d'assemblar el programa o d'algun detall concret, però no es pot posar el codi d'una subrutina o bucles sencers.

Competències

Les competències específiques que persegueix la PRÀCTICA són:

- [13] Capacitat per identificar els elements de l'estructura i els principis de funcionament d'un ordinador.
- [14] Capacitat per analitzar l'arquitectura i organització dels sistemes i aplicacions informàtics en xarxa.
- [15] Conèixer les tecnologies de comunicacions actuals i emergents i saber-les aplicar convenientment per dissenyar i desenvolupar solucions basades en sistemes i tecnologies de la informació.

Objectius

Introduir l'estudiant a la programació de baix nivell d'un computador, utilitzant el llenguatge assemblador de l'arquitectura Intel x86-64 i el llenguatge C.

Recursos

Podeu consultar els recursos de l'aula però no podeu fer ús intensiu del fòrum.

El material bàsic que podeu consultar és:

- Mòdul 6: Programació en assemblador (x86_64)
- Document "Entorn de treball"

La Pràctica: El Buscamines

La pràctica consisteix en implementar el joc del "BuscaMines" que consisteix en trobar on són les mines en un tauler de 10 x 10 caselles sense obrir cap casella que tingui una mina. Es poden marcar les caselles on creiem que hi ha una mina. Si s'obre una casella que no té mina s'indicarà quantes mines hi ha a les 8 caselles del voltant, amb aquesta informació hem de ser capaços de trobar on són totes les mines. Si s'obre una casella que té una mina, es perd la partida. El funcionament és semblant al "BuscaMines" de Windows.

La pràctica consta d'un programa en C, que us donem fet i NO HEU DE MODIFICAR, i un programa en assemblador que conté algunes subrutines ja fetes i altres que heu d'implementar vosaltres. Més avall trobareu la informació detallada sobre la implementació de la pràctica.

El fitxer C conté una versió completa de la pràctica per a que us serveixi de guia a l'hora d'implementar les subrutines en assemblador. També us permet executar el joc per veure com ha de funcionar.

La pràctica s'ha dividit en dues parts:

PRIMERA PART OBLIGATÒRIA:

Per a aquesta primera part us proporcionem dos fitxers: BMp1c.c i BMp1.asm. El codi C (BMp1c.c) no l'heu de modificar només heu d'implementar en assemblador en el fitxer MMp1.asm les funcionalitats necessàries.

Per a dividir la implementació del joc en cadascuna de les subrutines que cal realitzar i facilitar-vos la comprovació del seu funcionament, el programa C genera un menú principal amb 10 opcions:

- posCurScreen, crida a la subrutina posCurScreeenP1. Posiciona el cursor a la pantalla dins del tauler, en funció de l'índex de la matriu (indexMat), posició del cursor dins del tauler.
- showMines, crida a la subrutina showMinesP1. Converteix el valor del número de mines que queden per marcar, numMines, valor entre 0 i 99, a dos caràcters ASCII. Mostra a la part inferior del tauler les mines que queden per marcar, inicialment en queden 18.
- updateBoard, crida a la subrutina updateBoardP1. Actualitza el contingut del tauler de joc amb les dades de la matriu *marks* i el nombre de mines que queden per marcar cridant la subrutina showMinesP1.
- moveCursor, crida a la subrutina moveCursorP1. Actualitza la posició del cursor al tauler que tenim indicada amb la variable (indexMat), en funció a la tecla premuda que tenim a la variable (charac). Si es surt fora del tauler no actualitzar la posició del cursor.
- mineMarker, crida a la subrutina mineMarkerP1. Marca/desmarca una mina a la matriu (marks) a la posició actual del cursor, indicada amb la variable (indexMat).
- checkMines, crida a la subrutina checkMinesP1. Verifica si hem marcat totes les mines. Si numMines és 0, canvia l'estat del joc per a indicar que hem guanyat.
- Play Game, crida a la subrutina playP1. Permet provar totes les funcionalitats cridant les subrutines d'assemblador que s'han d'implementar en aquesta primera part, amb la tecla 'ESC' es pot sortir del joc i tornar al menú. Aquesta subrutina es dóna implementada.

- Play Game C, permet provar totes les funcionalitats cridant les funcions de C que us donem fetes en aquesta primera part, amb la tecla 'ESC' es pot sortir del joc i tornar al menú, **amb aquesta opció podeu veure el funcionament del joc**.
- Exit, finalitzar el programa.

Us recomanem que aneu desenvolupant el codi seguint l'ordre d'aquestes opcions del menú fins arribar a l'opció que permet jugar utilitzant totes les funcionalitats anteriors. Cada opció permet comprovar el funcionament de cada subrutina de forma independent.

En aquesta primera part no estarà del tot implementat el joc del BuscaMines. En la segona part opcional s'implementaran les funcionalitats necessàries per tenir un joc totalment funcional.

SEGONA PART OPCIONAL: En la segona part s'hauran d'implementar les funcionalitats addicionals necessàries per completar totes les funcionalitats del joc del BuscaMines.

A més a més, caldrà treballar amb el pas de paràmetres entre les diferents subrutines, modificant la implementació feta en la primera part.

Us proporcionarem també dos fitxers per a aquesta segona part: BMp2c.c i BMp2.asm. De forma semblant a la primera part, el codi C no l'haureu de modificar, només haureu d'implementar en assemblador noves funcionalitats i caldrà modificar les subrutines que haureu fet en la primera part per a treballar el pas de paràmetres entre les subrutines d'assemblador i també amb les funcions de C.

El 29 de novembre de 2019 us donarem els programes .c i .asm corresponents a la 2a part opcional.

Format i dates de lliurament

La **primera part** es pot lliurar abans de les **23:59 del dia 8 de novembre de 2019** per a obtenir una puntuació de pràctiques que pot arribar a una B. Si en aquesta data s'ha fet el lliurament de la primera part de manera satisfactòria es pot lliurar la **segona part** abans de les **23:59 del 13 de desembre de 2019** per a poder arribar a una puntuació de A en les pràctiques.

En canvi, si no s'ha pogut fer el primer lliurament o aquest primer lliurament no ha estat satisfactori es pot fer un **segon lliurament** abans de les **23:59 del 13 de desembre de 2019**. En aquesta segona data de lliurament es pot lliurar la primera part, per a obtenir una qualificació màxima de C+, o ambdues parts (la pràctica completa), per a obtenir una qualificació màxima de B.

Aquest esquema de lliuraments i qualificació es pot resumir en la següent taula.

Primer Lliurament 08-11-2019	**Primera part Superada**	**Primera part NO Superada** NO Presentat	**Primera part Superada**	**Primera part NO Superada** NO Presentat	**Primera part NO Superada** NO Presentat
Segon Lliurament 13-12-2019	**Segona part NO Superada** NO Presentat	**Primera part Superada**	**Segona part Superada**	**Primera part Superada** **Segona part Superada**	**Primera part NO Superada** NO Presentat
Nota Final Pràctica	**B**	**C+**	**A**	**B**	**D/N**

Els alumnes que no superin la PRÀCTICA tindran un suspens (qualificació 0-2) o N si no s'ha presentat res, en la nota final de pràctiques i amb aquesta nota no es pot aprovar l'assignatura, per aquest motiu la PRÀCTICA és obligatòria.

El lliurament s'ha de fer a través de l'aplicació **Lliurament i registre d'AC** de l'aula. S'ha de lliurar només un fitxer amb el codi assemblador ben comentat.

És molt important que el nom del fitxer tingui els vostres cognoms i nom amb el format:

cognom1_cognom2_nom_mP1.asm

Data límit Primer Lliurament:

Divendres, 8 de novembre de 2019 a les 23:59:59

Data límit Segon Lliurament:

Divendres, 13 de desembre de 2019 a les 23:59:59

Criteris de valoració

La pràctica ha de funcionar completament per a considerar-se superada, i cal implementar totes les funcionalitat demanades en l'enunciat, **l'opció del menú corresponent al joc complet en assemblador ha de funcionar correctament.**

Les altres opcions del menú són només per comprovar individualment cadascuna de les subrutines que s'han d'implementar.

No és suficient per aprovar la pràctica que les opcions corresponents a les subrutines individuals funcionin.

Donat que es tracta de fer un programa, seria bo recordar que les dues virtuts a exigir, per aquest ordre són:

- Eficàcia: que faci el que s'ha demanat i tal com s'ha demanat, és a dir, que tot funcioni correctament segons les especificacions donades. Si

les subrutines no tenen la funcionalitat demanada, encara que la pràctica funcioni correctament, es podrà considerar suspesa.

- Eficiència: que ho faci de la millor forma. Evitar codi redundant (que no faci res) i massa codi repetit (que el codi sigui compacte però clar). Fer servir modes d'adreçament adients: els que calguin. Evitar l'ús excessiu de variables i fer servir registres per emmagatzemar valors temporals.

IMPORTANT: Per a accedir als vectors en assemblador s'ha d'utilitzar adreçament relatiu o adreçament indexat: [marks+eax], [marks+edi]. No es poden utilitzar índexs amb valors fixos per accedir als vectors.

Exemple del que **NO** es pot fer:

```
mov BYTE [marks+0], 0
mov BYTE [marks+1], 0
mov BYTE [marks+2], 0
...
```

I repetir aquest codi molts cops.

Un altre aspecte important és la documentació del codi: que clarifiqui, doni ordre i estructura, que ajudi a entendre'l millor. No s'ha d'explicar que fa la instrucció (es dóna per suposat que qui la llegeix sap assemblador) si no que s'ha d'explicar perquè es fa servir una instrucció o grup d'instruccions (per fer quina tasca de més alt nivell, relacionada amb el problema que volem resoldre).

Implementació

Com ja hem dit, la pràctica consta d'una part de codi en C que us donem feta i **NO PODEU MODIFICAR** i un programa en assemblador que conté algunes subrutines ja implementades i les subrutines que heu d'implementar. Cada subrutina d'assemblador té una capçalera que explica que fa aquesta subrutina i com s'ha d'implementar, indicant les variables, funcions de C i subrutines d'assemblador que s'han de cridar.

No heu d'afegir altres variables o subrutines.

Per a ajudar-vos en el desenvolupament de la pràctica, en el fitxer de codi C trobareu implementat en aquest llenguatge les subrutines que heu de fer en assemblador per a que us serveixin de guia durant la codificació.

En el codi C es fan crides a les subrutines d'assemblador que heu d'implementar, però també trobareu comentades les crides a les funcions de C equivalents. Si voleu provar les funcionalitat fetes en C ho podeu fer traient el comentari de la crida de C i posant-lo en la crida a la subrutina d'assemblador.

Per exemple en l'opció 1 del menú fet en C hi ha el codi següent:

```
//=========================================================
posCurScreenP1();
//posCurScreenP1_C();
//=========================================================
```

El codi fa una crida a la subrutina d'assemblador updateBoardP1(), podem canviar el comentari i cridar a la funció de C.

```
//=========================================================
//posCurScreenP1();
posCurScreenP1_C();
//=========================================================
```

Recordeu tornar a deixar el codi com estava per a provar les vostres subrutines.

Les subrutines següents de la Primera Part ja estan implementades i NO les heu de modificar:

`gotoxyP1`
`printchP1`
`getchP1`
`playP1`

Les subrutines que cal implementar en assemblador per a la Primera Part són:

`posCurScreenP1`
`showMinesP1`
`updateBoardP1`
`moveCursorP1`
`mineMarkerP1`
`checkMinesP1`

Consulteu en el fitxer de codi assemblador (BMp1.asm) quin ha de ser el funcionament de cadascuna de les subrutines que heu d'implementar i quines variables utilitzen.