PRÀCTICA 1a part: Buscamines

Estructura de Computadors

Grau en Enginyeria Informàtica

feb 2017 - jun 2017

Estudis d'Informàtica, Multimèdia i Telecomunicació

EIMT.UOC.EDU

UOC 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 un joc del Buscamines.

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.

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 BMp1.asm les funcionalitats següents:

- ***posCurScreenP1***: Posiciona el cursor a la pantalla dins del tauler, en funció del vector *rowcol*, posició del cursor dins del tauler, (fila rowcol[0] i columna rowcol[1]). Per a calcular la posició del cursor a la pantalla cal utilitzar aquestes fórmules:

  rowScreen =(rowcol[0]*2)+7

  colScreen =(rowcol[1]*4)+7

- ***showMinesP1***: Converteix el valor del número de mines que queden per marcar *numMines* (valor entre 0 i 99) en dos caràcters ASCII. S'ha de dividir el valor *numMines* entre 10, el quocient representarà les desenes i el residu les unitats, i després s'han de convertir a ASCII sumant 48, carácter '0'. Mostra els dígits (caràcter ASCII) de les desenes a la fila 27, columna 23 de la pantalla i les unitats a la fila 27, columna 24, la posició s'indica a través de les variables *rowScreen* i *colScreen*.

- ***updateBoardP1***: Actualitza el contingut del tauler de joc amb les dades de la matriu *marks* i el nombre de mines que queden per marcar. S'ha de recórrer tota la matriu *marks*, per a cada element de la matriu posicionar el cursor a la pantalla i mostrar els caràcters de la matriu. Després mostrar el valor de *numMines* a la part inferior del tauler.

- ***moveCursorP1:*** Actualitza la posició del cursor en el tauler, que tenim en el vector *rowcol* (fila rowcol[0] i columna rowcol[1]), en funció de la tecla premuda, que tenim en la variable *charac* (i:amunt, j:esquerra, k:avall, l:dreta). Si es surt fora del tauler no actualitza la posició del cursor.

  rowcol[0] = rowcol[0] +/- 1

  rowcol[1] = rowcol[1] +/- 1

  No s'ha de posicionar el cursor a la pantalla.

- ***calcIndexMatP1***: Calcula el valor de l'índex per a accedir a les matrius *marks* i *mines* (10x10) que guardarem en la variable *indexMat* a partir dels valors de la posició actual del cursor dins del tauler, indicades pel vector *rowcol* (fila rowcol[0] i columna rowcol[1). Aquesta subrutina no té una funció en C equivalent.

- ***mineMarkerP1***: Marca/desmarca una mina en la matriu *marks* en la posició actual del cursor dins del tauler, indicada pel vector *rowcol* (fila

rowcol[0] i columna rowcol[1]). Si en aquella posició de la matriu *marks* hi ha un espai en blanc i no s'han marcat totes les mines, marquem una mina posant una 'M' i decrementem el número de mines que queden per marcar *numMines*, si en aquella posició de la matriu *marks* hi ha una 'M', hi posarem un espai (' ') i incrementarem el número de mines que queden per marcar *numMines*. Si hi ha un altre valor no canviarem res. Calcula el valor de l'índex, que es guarda en la variable *indexMat*, per a accedir a la matriu *marks* a partir dels valors indicats pel vector *rowcol* (fila rowcol[0] i columna rowcol[1]), cridant a la subrutina calcIndexP1. indexMat=(rowcol[0]*DimMatrix)+(rowcol[1])

No s'ha de mostrar la matriu, només cal actualitzar la matriu *marks* i la variable *numMines*.

El programa en C genera un menú principal amb 10 opcions:

- 1. posCurScreen, cridar a la subrutina posCurScreeenP1 per a provar el seu funcionament. Posiciona el cursor a la fila 1, columna 1 del tauler.
- 2. showMines, cridar a la subrutina showMinesP1 per a provar el seu funcionament. Mostra a la part inferior del tauler les mines que queden per marcar, inicialment en queden 18.
- 3. updateBoard, cridar a la subrutina updateBoardP1 per a provar el seu funcionament. 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.
- 4. moveCursor, cridar a la subrutina moveCursorP1 per a provar el seu funcionament. Posiciona el cursor a la fila 2, columna 2 cridant a la funció posCurScreenP1_C. Demana en quina direcció es vol moure el cursor (i:amunt, j:esquerra, k:avall, l:dreta), actualitza la posició del cursor, que tenim en el vector *rowcol* (fila rowcol[0] i columna rowcol[1]) en funció a la tecla premuda, que tenim en la variable *charac* cridant a la subrutina moveCursorP1. Després torna a posicionar el cursor a la nova posició cridant a la funció posCurScreenP1_C.
- 5. calcIndexMat, cridar a la subrutina calcIndexMatP1 per a provar el seu funcionament. Es mostra l'índex per a accedir a la fila 9, columna 9 de la matriu *marks* o *mines* cridant a la subrutina calcIndexMatP1.
- 6. mineMarker, cridar a la subrutina mineMarkerP1 per a provar el seu funcionament. Actualitza el contingut del tauler amb la matriu *marks* cridant a la funció updateBoardP1_C. Posiciona el cursor a la fila 3, columna 3 cridant a la funció posCurScreenP1_C. Demana que es premi la tecla 'm' per a marcar la mina. Marca / desmarca una mina en la matriu *marks* en la posició actual del cursor (fila 3, columna 3) cridant a la subrutina mineMarkerP1. Després torna a actualitzar el contingut del tauler amb la matriu *marks* cridant a la funció updateBoardP1_C.

- 7. checkMines, cridar a la subrutina checkMinesP1 que us donem feta per a provar el seu funcionament. Posa la variable *numMines* a zero i crida a la subrutina checkMines per a verificar si hem marcat totes les mines. Després mostra un missatge segons el valor de la variable *state* que s'ha actualitzat al cridar a la subrutina checkMinesP1.
- 8. Play Game, permet provar totes les funcionalitats cridant les subrutines d'assemblador que s'han d'implementar d'aquesta primera part. Amb la tecla 'ESC' s'ha de poder sortir del joc i tornar al menú.
- 9. Play Game C, permet provar totes les funcionalitats cridant les funcions de C que us donem fetes d'aquesta primera part. Amb la tecla 'ESC' s'ha de poder sortir del joc i tornar al menú. **Amb aquesta funció podeu veure el funcionament del joc**.
- 0. Exit, finalitzar el programa.

Us recomanem que aneu desenvolupant el codi seguin l'ordre d'aquestes opcions del menú fins arribar a l'opció 8 que permet jugar utilitzant totes les funcionalitats anteriors. Cada opció permet comprovar el funcionament de cada subrutina de forma independent.

En aquesta primera part el joc del Buscamines no estarà completament implementat i en realitat no podrem jugar una partida completa. En la segona part opcional s'implementaran les funcionalitats necessàries per tenir un joc totalment funcional.

Accés a les dades

Recordeu que les variables de tipus 'char' en assemblador s'han d'assignar a registres de tipus BYTE (1 byte): al, ah, bl, bh, cl, ch, dl, dh, sil, dil, r8b, r9b, ...

i les de tipus 'int' s'han d'assignar a registres de tipus DWORD (4 bytes): eax, ebx, ecx, edx, esi, edi, r8d, r9d, ...

A la pràctica treballarem amb matrius de 10x10 de tipus char, per tant, cada posició de la matriu ocuparà 1 byte, això fa que per passar d'una posició a la següent l'índex que utilitzem per accedir a la matriu s'incrementi de 1 en 1.

Per exemple, marks[0][0] serà BYTE[marks+0] i marks[0][1] serà BYTE[marks+1].

També utilitzarem un vector de 2 elements de tipus int, per tant, cada posició ocuparà 4 bytes, això fa que per passar d'una posició a la següent l'índex que utilitzem per accedir al vector s'incrementi de 4 en 4.

Per exemple, rowcol[0] serà DWORD[rowcol+0] i rowcol[1] serà DWORD[rowcol+4].

SEGONA PART OPCIONAL: En la segona part s'hauran d'implementar les funcionalitats addicionals necessàries per tenir un joc del Buscamines totalment funcional.

A més a més, caldrà treballar amb el pas de paràmetres entre les diferents subrutines i fer una correcte gestió de la pila, 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.

Format i dates de lliurament

La **primera part** es pot lliurar abans de les **23:59 del dia 21 d'abril de 2017** per 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 26 de maig de 2017** 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 26 de maig de 2017**. 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 21-04-2017	**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 26-05-2017	**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 (2-3 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 important que el nom del fitxer tingui els vostres cognoms i nom amb el format:

cognom1_cognom2_nom_P1.asm

Data límit Primer Lliurament:

Divendres, 21 d'abril de 2017 a les 23:59:59

Data límit Segon Lliurament:

Divendres, 26 de maig de 2017 a les 23:59:59

Criteris de valoració

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.

La pràctica ha de funcionar completament per a considerar-se superada, l'opció del 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.

IMPORTANT: Per a accedir als vectors en assemblador s'ha d'utilitzar adreçament relatiu o adreçament indexat: [marks+esi], [ebx+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:

```
//=======================================================
  rowcol[0]=1;  //Fila i columna on es posiciona el cursor
  rowcol[1]=1;
  posCurScreenP1();
  //posCurScreenP1_C();
//=======================================================
```

El codi fa una crida a la subrutina d'assemblador posCurScreenP1(), podem canviar el comentari i cridar a la funció de C.

```
//=======================================================
  rowcol[0]=1;  //Fila i columna on es posiciona el cursor
  rowcol[1]=1;
  //posCurScreenP1();
  posCurScreenP1_C();
//=======================================================
```

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

Subrutines que ja estan implementades i que no heu de modificar per a la Primera Part:

```
gotoxyP1
printchP1
getchP1
checkMinesP1
playP1
```

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

```
;;;;;
; Posiciona el cursor a la pantalla dins del tauler, en funció
; del vector (rowcol), posició del cursor dins del tauler,
; (fila rowcol[0] i columna rowcol[1]).
; Per a calcular la posició del cursor a la pantalla utilitzar
; aquestes fórmules:
; rowScreen =(rowcol[0]*2)+7
; colScreen =(rowcol[1]*4)+7
; Per a posicionar el cursor cridar a la subrutina gotoxyP1.
;
; Variables globals utilitzades:
; rowScreen: Fila de la pantalla on posicionem el cursor.
; colScreen: Columna de la pantalla on posicionem el cursor.
; rowcol   : vector amb la fila i la columna del cursor dins del tauler.
;;;;;  ;;;;;
posCurScreenP1:
```

```
;;;;;
; Converteix el valor del Número de mines que queden per marcar (numMines)
; (entre 0 i 99) a dos caràcters ASCII.
; S'ha de dividir el valor (numMines) entre 10, el quocient representarà
; les desenes i el residu les unitats, i després s'han de convertir
; a ASCII sumant 48, carácter '0'.
; Mostra els dígits (caràcter ASCII) de les desenes a la fila 27,
; columna 23 de la pantalla i les unitats a la fila 27, columna 24,
; (la posició s'indica a través de les variables rowScreen i colScreen).
; Per a posicionar el cursor cridar a la subrutina gotoxyP1 i per a mostrar
; els caràcters a la subrutina printchP1.
;
; Variables globals utilitzades:
; rowScreen: Fila de la pantalla on posicionem el cursor.
; colScreen: Columna de la pantalla on posicionem el cursor.
; numMines : Nombre de mines que queden per marcar.
; charac   : Caràcter a escriure a pantalla.
;;;;;
showMinesP1:


;;;;;
; Actualitzar el contingut del Tauler de Joc amb les dades de la matriu
; (marks) i el nombre de mines que queden per marcar.
; S'ha de recórrer tot la matriu (marks), i per a cada element de la matriu
; posicionar el cursor a la pantalla i mostrar els caràcters de la matriu.
; Després mostrar el valor de (numMines) a la part inferior del tauler.
; Per a posicionar el cursor cridar a la subrutina gotoxyP1, per a mostrar
; els caràcters a la subrutina printchP1 i per a mostrar (numMines)
; a la subrutina ShowMinesP1.
;
; Variables globals utilitzades:
; rowScreen: Fila de la pantalla on posicionem el cursor.
; colScreen: Columna de la pantalla on posicionem el cursor.
; charac   : Caràcter a escriure a pantalla.
; marks    : Matriu amb les mines marcades i les mines de les obertes.
;;;;;
updateBoardP1:


;;;;;
; Actualitzar la posició del cursor al tauler, que tenim al vector
; (rowcol) (fila rowcol[0] i columna rowcol[1]) en funció a la tecla
; premuda, que tenim a la variable (charac).
; Si es surt fora del tauler no actualitzar la posició del cursor.
; (i:amunt, j:esquerra, k:avall, l:dreta)
; ( rowcol[0] = rowcol[0] +/- 1 ) ( rowcol[1] = rowcol[1] +/- 1 )
; No s'ha de posicionar el cursor a la pantalla.
;
; Variables globals utilitzades:
; rowcol   : vector amb la fila i la columna del cursor dins del tauler.
; charac   : Caràcter llegit de teclat.
;;;;;
moveCursorP1:


;;;;;
; Calcular el valor de l'índex per a accedir a la matriu (marks) i (mines)
; (10x10) que guardarem a la variable (indexMat) a partir dels valors de
; la posició actual del cursor. Indicades pel vector: (rowcol).
; (rowcol: vector amb la fila rowcol[0] i la columna rowcol[1] del
; cursor dins del tauler.)
;
; Aquesta subrutina no té una funció en C equivalent.
;
; Variables globals utilitzades:
; rowcol   : vector amb la fila i la columna del cursor dins del tauler.
; indexMat : índex per a accedir a les matrius mines i marks.
;;;;;
```

```
calcIndexP1:


;;;;;
; Marcar/desmarcar una mina a la matriu (marks) a la posició actual del
; cursor, indicada pel vector (rowcol) (fila (rowcol[0]) i columna
; (rowcol[1]) del cursor dins del tauler).
; Si en aquella posició de la matriu (marks) hi ha un espai en blanc i
; no s'han marcat totes les mines, marquem una mina posant una 'M' i
; decrementem el número de minas que queden per marcar (numMines),
; si en aquella posició de la matriu (marks) hi ha una 'M', hi posarem
; un espai (' ') i incrementarem el número de mines que quedan per
; marcar (numMines).
; Si hi ha un altre valor no canviarem res.
; Calcular el valor de l'índex, que es guarda a la variable (indexMat),
; per a accedir a la matriu (marks) a partir dels valors indicats pel
; vector (rowcol)(rowcol: vector amb la fila rowcol[0] i la columna
; rowcol[1] del cursor dins del tauler), cridant a la subrutina
; calcIndexP1. (indexMat=(rowcol[0]*DimMatrix)+(rowcol[1]))
; No s'ha de mostrar la matriu, només actualitzar la matriu (marks)
; i la variable (numMines).
;
; Variables globals utilitzades:
; indexMat : índex per a accedir a la matriu marks.
; marks    : Matriu amb les mines marcades i les mines de les obertes.
; numMines : nombre de mines que queden per marcar.
;;;;;
mineMarkerP1:
```