PRÀCTICA 2a part: Buscamines

Estructura de Computadors

Grau en Enginyeria Informàtica

Feb 2107 - 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 implementar 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 rutines ja fetes i altres que heu d’implementar vosaltres. Més avall trobareu la informació detallada sobre la implementació de la pràctica.

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

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'heu de modificar, només heu 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, també cal fer a cada subrutina una correcte gestió de la pila guardant a l’inici els registres que es modifiquen, i no son paràmetres de retorn, i restaurant el seu valor abans de sortir.

Per aquesta segona part cal implementar les funcionalitats següents:

- ***serachMinesP2:*** Obrir casella amb la tecla ‘ESPAI’. Mirar quantes mines hi ha al voltant de la posició actual del cursor, indicada pel vector (rowcol) que es rep com a paràmetre (fila (rowcol[0]) i columna (rowcol[1])), de la matriu (mines). Si a la posició actual de la matriu (marks) hi ha un espai (' ') i a la matriu (mines) hi ha una mina ('*') , canviar l'estat (state), que es rep com a paràmetre, a 3 "Explosió", per a sortir. Sinó, comptar quantes mines hi ha al voltant de la posició actual i actualitzar la posició de la matriu (marks) amb el nombre de mines. Sinó hi ha un espai, vol dir que hi ha una mina marcada ('M') o la casella ja s'ha obert anteriorment i no cal fer res.

- ***checkEndP2***: Verificar si hem marcat totes les mines (numMines=0), que es rep com a paràmetre (aquesta verificació ja es feia a la 1a part de la pràctica amb la subrutina checkMinesP1) i si hem obert o marcat amb una mina totes les altres caselles i no hi ha cap espai en blanc (' ') a la matriu (marks), si és així, canviar l'estat (state) que es rep com a paràmetre, a 2 "Guanya". Comprovar si ha finalitzat la partida i hem guanyat el joc: s'han marcat totes les mines, *numMines*=0, i s'han obert la resta de caselles del tauler, no queden espais en blanc en la matriu *marks*.

- ***posCurScreenP2, showMinesP2, updateBoardP2, moveCursorP2, calcIndexP2, mineMarkerP2***: Aquestes subrutines tenen una funcionalitat molt semblant a les subrutines implementades a la P1, la tasca principal es modificar el pas de paràmetres i la gestió de la pila.

En aquesta segona part el programa en C també genera un menú principal amb 10 opcions:

- 1. posCurScreen, cridar la subrutina posCurScreeenP2 per a provar el seu funcionament. Posiciona el cursor a la fila 1, columna 1 del tauler.
- 2. showMines, cridar la subrutina showMinesP2 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 la subrutina updateBoardP2 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 showMinesP2.
- 4. moveCursor, cridar la subrutina moveCursorP2 per a provar el seu funcionament. Posiciona el cursor a la fila 2, columna 2 cridant a la funció posCurScreenP2_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 al vector (rowcol) (fila rowcol[0] i columna rowcol[1]) en funció a la tecla premuda, que tenim a la variable (charac) cridant a la subrutina moveCursorP2. Després torna a posicionar el cursor a la nova posició cridant a la funció posCurScreenP2_C.
- 5. mineMarker, cridar la subrutina mineMarkerP2 per a provar el seu funcionament. Actualitza el contingut del tauler amb la matriu marks cridant la funció updateBoardP2_C. Posiciona el cursor a la fila 5, columna 4 cridant a la funció posCurScreenP2_C. Demana que es premi la tecla ‘m’ per a marcar la mina. Marcar/desmarcar una mina a la matriu (marks) a la posició actual del cursor (fila 3, columna 3) cridant la subrutina mineMarkerP2. Després torna a actualitzar el contingut del tauler amb la matriu marks cridant la funció updateBoardP2_C.
- 6. searchMines, cridar la subrutina searchMinesP2 per a provar el seu funcionament. Actualitza el contingut del tauler amb la matriu marks cridant la funció updateBoardP2_C. Posiciona el cursor a la fila 5 columna 4 del tauler cridant la funció posCurScreeP2_C on obrim la casella. Demana que es premi la tecla ‘ESPAI’ per a obrir la casella. Mira quantes mines hi ha al voltant de la posició actual del cursor, indicada pel vector (rowcol), a la matriu (mines). Si a la posició actual de la matriu (marks) hi ha un espai (' ') mirar si a la matriu (mines) hi ha una mina ('*'). Si hi ha una mina canviar l'estat (state), que es rep com a paràmetre a 3 "Explosió", per a sortir. Sinó, compta quantes mines hi ha al voltant de la posició actual i actualitzar la posició de la matriu (marks) amb el nombre de mines. Si no hi ha un espai, vol dir que hi ha una mina marcada ('M') o la casella ja s'ha obert (hi ha el nombre

de mines que ja s'ha calculat anteriorment), no fer res cridant la subrutina searcMinesP2. Després torna a actualitzar el contingut del tauler amb la matriu marks cridant la funció updateBoardP2_C i mostra un missatge segons el valor de la variable state cridant la funció printMessageP2_C.

- 7. checkEnd, cridar la subrutina checkEndP2 per a provar el seu funcionament. Actualitza el contingut del tauler amb la matriu marks cridant la funció updateBoardP2_C. Verificar si hem marcat totes les mines (numMines=0) i si hem obert o marcat amb una mina totes les altres caselles i no hi ha cap espai en blanc (' ') a la matriu (marks), si és així, canviar l'estat (state) a 2 "Guanya" cridant la subrutina checkEndP2. Després mostra un missatge segons el valor de la variable state cridant la funció printMessageP2_C.
- 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.

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, ..., r15b i les de tipus 'int' s'han d'assignar a registres de tipus DWORD (4 bytes): eax, ebx, ecx, edx, esi, edi, r8d, r9d, ..., r15d.

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].

Format i data de lliurament

Aquesta **segona part** s'ha de lliurar abans de les **23:59 del 26 de maig de 2017**. Si en el primer lliurament es va superar la 1a part es pot arribar a una puntuació de A en les pràctiques.

En canvi, si no es va fer el primer lliurament o aquest primer lliurament no va ser satisfactori es pot lliurar la primera part juntament amb aquesta 2a part, 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 molt important que el nom del fitxers tingui els vostres cognoms i nom amb el format:

cognom1_cognom2_nom_p2.asm

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 (opció 8) 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: [mines+esi], [ebx+edi]. No es poden utilitzar índexs amb valors fixos per accedir als vectors.

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

```
mov BYTE [mines+0], 0
mov BYTE [mines+1], 0
mov BYTE [mines+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 d'implementar.

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;
  posCurScreenP2(rowcol);
  //posCurScreenP2_C(rowcol);
//=====================================================
```

El codi fa una crida a la subrutina d'assemblador posCurScreenP2(rowcol), 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;
  //posCurScreenP2(rowcol);
  posCurScreenP2_C(rowcol);
//=====================================================
```

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

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

`gotoxyP2`
`printchP2`
`getchP2`
`playP2`

Les subrutines de la Primera Part que s'han de modificar per a implementar el pas de paràmetres, fer la gestió de la pila i ampliar alguna funcionalitat en aquesta segona part són:

`posCurScreenP2`
`showMinesP2`
`updateBoardP2`
`moveCursorP2`
`calcIndexP2`
`mineMarkerP2`

Les subrutines noves que cal implementar en assemblador per a la Segona Part implementant el pas de paràmetres i fent la gestió de la pila són:

```
;;;;;
; Obrir casella. Mirar quantes mines hi ha al voltant de la posició
; actual del cursor, indicada pel vector (rowcol) que es rep com a
; paràmetre (fila (rowcol[0]) i columna (rowcol[1])), de la
; matriu (mines).
; Si a la posició actual de la matriu (marks) hi ha un espai (' ')
;   Mirar si a la matriu (mines) hi ha una mina ('*').
;   Si hi ha una mina canviar l'estat (state), que es rep com a
;     paràmetre a 3 "Explosió", per a sortir.
;       Sinó, comptar quantes mines hi ha al voltant de la posició
;     actual i actualitzar la posició de la matriu (marks) amb
;     el nombre de mines (caràcter ASCII del valor, per fer-ho, cal
;     sumar 48 ('0') al valor).
; Sinó hi ha un espai, vol dir que hi ha una mina marcada ('M') o la
; casella ja s'ha obert (hi el nombre de mines que ja s'ha calculat
; anteriorment), no fer res.
; Retornar l'estat del joc actualitzat.
; No s'ha de mostrar la matriu.
;
; Variables globals utilitzades:
; marks  : Matriu amb les mines marcades i les mines de les obertes.
; mines  : Matriu on posem les mines.

; Paràmetres d'entrada :
; rdi      : Adreça del vector de 2 posicions on guardem la fila
;            (rowcol[0]) i la columna (rowcol[1]) del cursor dins del tauler.
; rsi(esi) : Estat del joc.
;
; Paràmetres de sortida:
; rax(eax) : Estat del joc.
;;;;;
searchMinesP2:
```

```
;;;;;
; Verificar si hem marcat totes les mines (numMines=0), que es rep com
; a paràmetre i hem obert o marcat amb una mina totes  les altres
; caselles i no hi ha cap espai en blanc (' ') a la matriu (marks),
; si és així, canviar l'estat (state) que es rep com a paràmetre, a
; 2 "Guanya".
; Retornar l'estat del joc actualitzat.
;
; Variables globals utilitzades:
; marks    : Matriu amb les mines marcades i les mines de les obertes.
;
; Paràmetres d'entrada :
; rdi(edi) : Nombre de mines que queden per marcar.
; rsi(esi) : Estat del joc.
;
; Paràmetres de sortida:
; rax(eax) : Estat del joc.
;;;;;
checkEndP2:
```