4 IN A ROW

FONAMENTS DELS COMPUTADORS - NIVELL AVANÇAT

AUTORS:
ALBERT CAIRE RODRÍGUEZ
NIU: 1702574
MARÍA FLORIDO MUÑIZ
NIU: 1710768
EDUARDO PÉREZ MOTATO
NIU: 1709992

30 DE DESEMBRE DE 2023

Índex

1 Descripció

Aquest projecte tracta de crear un 4 en ratlla en ensamblador. Per això hem fet servir un archiu `.c` ja creat per l'apartat gràfic, del qual no parlarem, i unes subrutines dintre del `.asm`. Aquest projecte ha sigut subdividit en nivells de dificultat i per allò a l'apartat subrutines està separat per nivells.

2 Objectius

El joc del 4 en ratlla té com a objectiu principal col·locar quatre fitxes iguals en línia, ja sigui horitzontal, vertical o diagonal, abans que l'adversari ho faci. Això implica prevenir l'altra persona de formar aquesta línia i, al mateix temps, crear estratègies per aconseguir la teva.

3 Variables utilitzades

- **DWORD**`[colScreen]` (4 *bytes*): Columna on volem posicionar el cursor a la pantalla.
- **DWORD**`[rowScreen]` (4 *bytes*): Fila on volem posicionar el cursor a la pantalla.
- **DWORD**`[rowScreenIni]` (4 *bytes*): Fila de la primera posició de la matriu a la pantalla.
- **DWORD**`[colScreenIni]` (4 *bytes*): Columna de la primera posició de la matriu a la pantalla.
- **DWORD**`[player]` (4 *bytes*): Variable que indica el jugador al que correspon tirar.
- **BYTE**`[mBoard]` (1 *byte*): Matriu 6x7 on tenim les dades del taulell.
- **BYTE**`[tecla]` (1 *byte*): Variable on s'emmagatzema la tecla pitjada en codi ascii.
- **BYTE**`[colCursor]` (1 *byte*): Variable que indica la columna en la que es troba el cursor.
- **DWORD**`[row]` (4 *bytes*): Fila per a accedir a la matriu mBoard.
- **BYTE**`[col]` (1 *byte*): Columna per a accedir a la matriu mBoard.
- **DWORD**`[pos]` (4 *bytes*): Índex per a accedir a la matriu mBoard.
- **DWORD**`[inaRow]` (4 *bytes*): Comptador per a saber el nombre de fitxes en ratlla.
- **DWORD**`[row4Complete]` (4 *bytes*): Indicador de si hem arribat a 4 fitxes en ratlla o no.

4 Subrutines

4.1 Nivell bàsic

Aquest nivell tracta de posar a punt totes les subrutines base per aixi tindre un mínim de funcionalitat.

4.1.1 showCursor

```
246 showCursor:
247     push rbp
248     mov rbp, rsp
249 
250     push rax
251     push rbx
252     mov eax, 0
253     mov ebx, 0
254 
255     mov eax, DWORD [RowScreenIni]
256     mov DWORD [rowScreen], eax
257     mov al, BYTE [colCursor]
258     sub eax, 'A'
259     shl eax, 2
260     add eax, DWORD [ColScreenIni]
261     mov DWORD [colScreen], eax
262 
263     call gotoxy
264 
265     pop rbx
266     pop rax
267 
268     mov rsp, rbp
269     pop rbp
270     ret
```

Aquesta subrutina mostra la localització del cursor en pantalla. Per allò, fa servir `rowScreen`, `colScreen` i els seus corresponents `rowScreenIni`, `colScreenIni`. En aquesta subrutina no va haver cap dificultat.

4.1.2 showPlayer

```
294 showPlayer:
295     push rbp
296     mov rbp, rsp
297 
298     push rax
299 
300     mov eax, DWORD [player]
301     add eax, 48
302     mov DWORD [rowScreen], 23
303     mov DWORD [colScreen], 20
304 
305     call gotoxy
306     mov dil, al
307     call printch
308 
309     pop rax
310 
311     mov rsp, rbp
312     pop rbp
313     ret
```

Aquesta subrutina mostra el jugador que té que jugar a la posició de la pantalla on està. Aquesta subrutina no va comportar cap problema.

4.1.3 showBoard

```
337 showBoard:
338     push rbp
339     mov rbp, rsp
340 
341     push rsi
342     push rdi
343     push rax
344 
345     mov eax, 0
346     mov rdi, 0
347     mov rsi, 0
348 
349     mov DWORD [rowScreen], 8
350     buclefilas:
351         cmp DWORD [rowScreen], 20
352         jge fi_showBoard
353         add DWORD [rowScreen], 2
354         mov DWORD [colScreen], 4
355     buclecolumnas:
356         cmp DWORD [colScreen], 32
357         jge buclefilas
358         add DWORD [colScreen], 4
359         mov dil, BYTE [mBoard + eax]
360         inc eax
361         call gotoxy
362         call printch
363         jmp buclecolumnas
364 
365     fi_showBoard:
366 
367     call showPlayer
368 
369     pop rax
370     pop rdi
371     pop rsi
372 
373     mov rsp, rbp
374     pop rbp
375     ret
```

Aquesta subrutina ens mostra el tauler amb les fitxes que hagim col·locat. Aquesta va ser la tasca que més dificultats ens va comportar el nivell bàsic i hi vam trobar dos problemes principals. El primer va ser un error de numeració de files i columnes que al calcular malament el marge feia que col·loquèssim punts fora i el segon va ser que no accediem per iteracció a la matriu `mBoard` i s'omplia de `'.'` malgrat tindre fitxes col·locades. El

primer el vam arreglar dibuixant la matriu en paper per imaginar-ho millor i el segon comprovant-ho tot fins que ens vam adonar de que no havíem fet us de `mBoard`.

4.1.4 moveCursor

```
406 moveCursor:
407     push rbp
408     mov rbp, rsp
409     call showCursor
410     bucle_moveCursor:
411         call getch
412         cmp BYTE [tecla], 'j'
413         je moveLeft
414         cmp BYTE [tecla], 'k'
415         je moveRight
416         cmp BYTE [tecla], '␣'
417         je fi_moveCursor
418         cmp BYTE[tecla], 27
419         je fi_moveCursor
420         jmp bucle_moveCursor
421 
422     moveLeft:
423         cmp BYTE [colCursor], 'A'
424         je fi_moveCursor
425         dec BYTE [colCursor]
426         jmp fi_moveCursor
427 
428     moveRight:
429         cmp BYTE [colCursor], 'G'
430         je fi_moveCursor
431         inc BYTE [colCursor]
432 
433     fi_moveCursor:
434         call showCursor
435 
436     mov rsp, rbp
437     pop rbp
438     ret
```

Aquesta subrutina permet moure el cursor una sola posició a esquerra o a dreta utilitzant les tecles `'j'` i `'k'` respectivament. No ens va representar molta dificultat donat que aquesta tasca només és com una prèvia a `moveCursorContinous`.

4.1.5 moveCursorContinuous

```
454 moveCursorContinuous:
455     push rbp
456     mov rbp, rsp
457 
458     bucle_moveCursorContinuous:
459         call moveCursor
460         cmp BYTE [tecla], ' '
461         je fi_moveCursorContinuous
462         cmp BYTE [tecla], 27
463         je fi_moveCursorContinuous
464         jmp bucle_moveCursorContinuous
465 
466     fi_moveCursorContinuous:
467 
468     mov rsp, rbp
469     pop rbp
470     ret
```

Aquesta subrutina utilitza `moveCursor` però elimina la limitació de que el moviment sigui només d'una posició. Vam trobar problemes amb evitar que ens ensortíssim de la matriu i el cursor arroseguès el marge de la matriu. Vam solucionar-ho afegint restriccions de comprobació que no permetessin al cursor arribar al marge dret.

4.1.6 calcIndex

```
485 calcIndex:
486     push rbp
487     mov rbp, rsp
488 
489     push rax
490     push rbx
491 
492     mov eax, DWORD [row]
493     imul eax, 7
494     mov bl, BYTE [col]
495     sub bl, 'A'
496     add eax, ebx
497     mov DWORD [pos], eax
498 
499     pop rbx
500     pop rax
501 
502     mov rsp, rbp
503     pop rbp
504     ret
```

Aquesta subrutina ens permet accedir a cada component de la matriu mBoard via la fórmula canònica del tipus $i \times m + j$, on i és la fila a què es vol accedir, `DWORD[row]`; m és la mida de les files (7, ja que `mBoard` és una matriu 6×7); i j és la columna a què es vol accedir, `BYTE[col]` (es passa de forma alfabètica a forma numèrica restant-li el valor ascii de `'A'`). No van sorgir problemes majors en la realització d'aquesta subrutina.

4.1.7 putPiece

```
putPiece:
    push rbp
    mov rbp, rsp

    push rax
    push rbx

    mov eax, 0
    mov ebx, 0

    call showBoard
    call moveCursorContinuous

    cmp BYTE [tecla], ' '
    jne fi_putPiece

    mov al, BYTE [colCursor]
    mov BYTE [col], al
    mov DWORD [row], 5

    bucle_putPiece:
        call calcIndex
        mov eax, DWORD [pos]
        mov bl, BYTE [mBoard + eax]
        cmp bl, '.'
        je colocacion
        cmp DWORD [row], 0
        jl fi_putPiece
        dec DWORD [row]
        jmp bucle_putPiece

    colocacion:
        mov BYTE [mBoard + eax], 'X'
        call showBoard

    fi_putPiece:

    pop rbx
    pop rax

    mov rsp, rbp
    pop rbp
    ret
```

Aquesta subrutina permet col·locar una fitxa a `mBoard`. Quan es prem l'espai, es col·loca la fitxa tot controlant que no hi hagi una fitxa existent.

Un problema que vam tenir va ser que aquesta subrutina, en un principi, no comprovava que la columna es trobés plena i això resultava en "desbordaments". Ho vam solucionar programant millor el desbordaments.

4.2 Nivell mig

Aquest nivell implementa mes funcionalitats, la posibilitat de dos jugadors i una versió del joc força simplificat pero funcional.

4.2.1 checkRow

```
checkRow:
    push rbp
    mov rbp, rsp

    push rax
    push rbx

    mov rax, 0
    mov rbx, 0
    mov DWORD [inaRow], 0

    mov eax, DWORD [pos]
    mov bl, BYTE [mBoard + eax]
    bucle_vertical:
        cmp eax, 42
        jge fi_vertical
        cmp bl, BYTE [mBoard + eax]
        jne fi_vertical
        add eax, 7
        inc DWORD [inaRow]
        cmp DWORD [inaRow], 4
        jne bucle_vertical
        mov DWORD [row4Complete], 1

    fi_vertical:
    pop rbx
    pop rax

    mov rsp, rbp
    pop rbp
    ret
```

Aquesta subrutina comprova si algun dels dos jugadors ha guanyat o no. Només ho comprova en vertical així que no té una dificultat excesiva donat que aquesta tasca és també una prèvia de la que porta el mateix nom al nivell avançat. Per allò, no van sortir problemes.

4.2.2 putPiece

```
770 putPiece:
771     push rbp
772     mov rbp, rsp
773 
774     push rax
775     push rbx
776     mov eax, 0
777     mov ebx, 0
778     call showBoard
779     call moveCursorContinuous
780     cmp BYTE [tecla], '␣'
781     jne fi_putPiece
782 
783     mov al, BYTE [colCursor]
784     mov BYTE [col], al
785     mov DWORD [row], 5
786 
787     bucle_putPiece:
788         call calcIndex
789         mov eax, DWORD [pos]
790         mov bl, BYTE [mBoard + eax]
791         cmp bl, '.'
792         je colocacion
793         cmp DWORD [row], 0
794         jl fi_putPiece
795         dec DWORD [row]
796         jmp bucle_putPiece
797 
798     colocacion:
799         cmp DWORD [player], 1
800         jne jugador2
801         mov BYTE [mBoard + eax], '0'
802         jmp fi_putPiece
803         jugador2:
804             mov BYTE [mBoard + eax], 'X'
805 
806     fi_putPiece:
807         call showBoard
808         call checkRow
809 
810     pop rbx
811     pop rax
812 
813     mov rsp, rbp
814     pop rbp
815     ret
```

Aquesta subrutina és una millora de la del nivell bàsic. Aquesta versió controla el jugador actual per tal de mostrar per pantalla la fitxa `'O'` o `'X'`, segons si és el jugador 1 o el 2, respectivament. No van sorgir problemes majors en la realització d'aquesta subrutina.

4.2.3 put2Players

```
841 put2Players:
842     push rbp
843     mov rbp, rsp
844 
845     mov DWORD [player], 1
846 
847     bucle_put2Players:
848         call showPlayer
849         call putPiece
850         cmp BYTE [tecla], ' '
851         jne fi_put2Players
852         cmp DWORD [row4Complete], 1
853         je fi_put2Players
854         cmp DWORD [player], 2
855         je fi_put2Players
856         mov DWORD [player], 2
857         jmp bucle_put2Players
858 
859     fi_put2Players:
860 
861     mov rsp, rbp
862     pop rbp
863     ret
```

Aquesta subrutina permet una jugada, formada per un torn per cada jugador. Es força que el primer torn el jugui el jugador 1 i es controla que la subrutina finalitzi en el cas que el jugador guanyi en el seu torn (no podem permetre que el jugador 2 jugui si ja s'ha acabat el joc), així com que el jugador 2 jugui només un cop (sortim del bucle si aquest ja ha jugat un cop i/o si ha aconseguit guanyar, òbviament). No van sorgir problemes majors en la realització d'aquesta subrutina.

4.2.4 Play

```
880 Play:
881     push rbp
882     mov rbp, rsp
883 
884     bucle_Play:
885         call put2Players
886         cmp BYTE [tecla], '␣'
887         jne fi_Play
888         cmp DWORD [row4Complete], 1
889         je fi_Play
890         jmp bucle_Play
891 
892     fi_Play:
893 
894     mov rsp, rbp
895     pop rbp
896     ret
```

Aquesta subrutina implementa la jugabilitat. Primer crida a put2Players, després comprova l'ultima tecla pitjada, si aquesta no es un espai surt (ja que per força haura sigut un ESC) i comprova el valor de `DWORD[row4Complete]`, en cas de ser 1 ha guanyat i per allò surt. Si no es torna el bucle. No hi van haver problemes a aquesta subrutina.

4.3 Nivell avançat

Aquest nivell implementa les més opcions del `checkRow`.

4.3.1 checkRow

```
524 checkRow:
525     push rbp
526     mov rbp, rsp
527 
528     push rax
529     push rbx
530     push rcx
531     push rdx
532 
533     mov rax, 0
534     mov rcx, 0
535     mov rbx, 0
536     mov rdx, 0
537     mov DWORD [inaRow], 0
538     mov ecx, DWORD [pos]
539     mov bl, BYTE [mBoard + ecx]
540     cmp bl, '.'
541     je fi_vertical
542 
543     bucle_horizontal_adelante:
544         cmp bl, BYTE [mBoard + ecx]
545         jne fi_bucle_horizontal_adelante
546         inc ecx
547         inc DWORD [inaRow]
548         cmp DWORD [inaRow], 4
549         je fi_row4
550         mov eax, ecx
551         push rcx
552         mov rdx, 0
553         mov ecx, 7
554         div ecx
555         pop rcx
556         cmp edx, 0
557         je fi_bucle_horizontal_adelante
558         jmp bucle_horizontal_adelante
559 
560     fi_bucle_horizontal_adelante:
561         mov ecx, DWORD [pos]
562         dec ecx
563 
564     bucle_horizontal_atras:
565         cmp ecx, 0
```

```
566         jl fi_bucle_horizontal_atras
567         cmp bl, BYTE [mBoard + ecx]
568         jne fi_bucle_horizontal_atras
569         dec ecx
570         inc DWORD [inaRow]
571         cmp DWORD [inaRow], 4
572         je fi_row4
573         mov eax, ecx
574         push rcx
575         mov rdx, 0
576         mov ecx, 7
577         div ecx
578         pop rcx
579         cmp edx, 6
580         je fi_bucle_horizontal_atras
581         jmp bucle_horizontal_atras
582
583     fi_bucle_horizontal_atras:
584         mov ecx, DWORD [pos]
585         mov DWORD [inaRow], 0
586
587     bucle_diagonal_adelante:
588         cmp ecx, 42
589         jge fi_bucle_diagonal_adelante
590         cmp bl, BYTE [mBoard + ecx]
591         jne fi_bucle_diagonal_adelante
592         add ecx, 8
593         inc DWORD [inaRow]
594         cmp DWORD [inaRow], 4
595         je fi_row4
596         mov eax, ecx
597         push rcx
598         mov rdx, 0
599         mov ecx, 7
600         div ecx
601         pop rcx
602         cmp edx, 0
603         je fi_bucle_diagonal_adelante
604         jmp bucle_diagonal_adelante
605
606     fi_bucle_diagonal_adelante:
607         mov ecx, DWORD [pos]
608         sub ecx, 8
609
610     bucle_diagonal_atras:
611         cmp ecx, 0
612         jl fi_bucle_diagonal_atras
613         cmp bl, BYTE [mBoard + ecx]
614         jne fi_bucle_diagonal_atras
```

```
615         sub ecx, 8
616         inc DWORD [inaRow]
617         cmp DWORD [inaRow], 4
618         je fi_row4
619         mov eax, ecx
620         push rcx
621         mov rdx, 0
622         mov ecx, 7
623         div ecx
624         pop rcx
625         cmp edx, 6
626         je fi_bucle_diagonal_atras
627         jmp bucle_diagonal_atras
628
629     fi_bucle_diagonal_atras:
630         mov ecx, DWORD [pos]
631         mov DWORD [inaRow], 0
632
633     bucle_antidiagonal_adelante:
634         cmp ecx, 42
635         jge fi_bucle_antidiagonal_adelante
636         cmp bl, BYTE [mBoard + ecx]
637         jne fi_bucle_antidiagonal_adelante
638         add ecx, 6
639         inc DWORD [inaRow]
640         cmp DWORD [inaRow], 4
641         je fi_row4
642         mov eax, ecx
643         push rcx
644         mov rdx, 0
645         mov ecx, 7
646         div ecx
647         pop rcx
648         cmp edx, 6
649         je fi_bucle_antidiagonal_adelante
650         jmp bucle_antidiagonal_adelante
651
652     fi_bucle_antidiagonal_adelante:
653         mov ecx, DWORD [pos]
654         sub ecx, 6
655
656     bucle_antidiagonal_atras:
657         cmp ecx, 0
658         jl fi_bucle_antidiagonal_atras
659         cmp bl, BYTE [mBoard + ecx]
660         jne fi_bucle_antidiagonal_atras
661         sub ecx, 6
662         inc DWORD [inaRow]
663         cmp DWORD [inaRow], 4
```

```
664         je fi_row4
665         mov eax, ecx
666         push rcx
667         mov rdx, 0
668         mov ecx, 7
669         div ecx
670         pop rcx
671         cmp edx, 0
672         je fi_bucle_antidiagonal_atras
673         jmp bucle_antidiagonal_atras
674
675     fi_bucle_antidiagonal_atras:
676         mov ecx, DWORD [pos]
677         mov DWORD [inaRow], 0
678
679     bucle_vertical:
680         cmp ecx, 42
681         jge fi_vertical
682         cmp bl, BYTE [mBoard + ecx]
683         jne fi_vertical
684         add ecx, 7
685         inc DWORD [inaRow]
686         cmp DWORD [inaRow], 4
687         jne bucle_vertical
688
689     fi_row4:
690     mov DWORD [row4Complete], 1
691
692     fi_vertical:
693
694     pop rdx
695     pop rcx
696     pop rbx
697     pop rax
698
699     mov rsp, rbp
700     pop rbp
701     ret
```

Aquesta subrutina és una millora de la del nivell mig. Aquesta versió no només controla la vertical, horitzontal, diagonal i antidiagonal, sinó que ho fa independentment d'on s'hagi col·locat la última fitxa (al principi de la fila, al final o al mig). Van sorgir problemes diversos i tots eren relacionats amb la manera en què es comprovava si s'havia fet 4 en ratlla. Es va solucionar realitzant una documentació sobre la instrucció **div** (per tal d'aturar el bucle en cas de passar-se de la fila de la matriu `mBoard`) i reordenant el codi (així com afegint comprovacions a `checkRow`).

5 Altres

Aquest grup ha tingut la sort de comptar amb un integrant del grup que ja tenia coneixements previs sobre programació, però en altre cas ens hauria sigut pràcticament impossible arribar al nivell avançat en els termes de temps establerts. Tot i que el seguiment proporcionat pels professors durant les pràctiques ens ha sigut de gran ajuda, la introducció a les pràctiques es va fer de forma massa abrupta per a aquells que no haguèssim fet res més d'ensamblador que els codis de classe. Finalment ens agradaria mencionar que tot i que ha sigut un repte, hem gaudit prou de les pràctiques i que amb una introducció més detallada o uns enunciats més esclaridors seria ideal.