Estructura de Computadors II

Segona pràctica

· Bosch Matas, Francisco 43205369F

· Miró Barceló, Oriol 43182236N

- Enunciat

La segona pràctica consisteix en la programació d'un joc que faci ús d'un gestor de tasques que usa programació cooperativa.

Aquesta pràctica consta de dues parts:

La primera part consisteix en la programació del gestor multitasca i dels altres programes que l'acompanyen, que són el gestor de memòria dinàmica i la llista d'identificador disponibles.

El gestor de memòria dinàmica (*dmm.X68*) conté tres subrutines principals: *DMM_INIT*, que inicia la llista d'apuntadors a memòria disponible; *DMM_ALLOCATE*, que retorna un punter a memòria disponible, i *DMM_RELEASE*, que allibera el tros de memòria donat.

La llista d'identificador (*idpool.X68*) conté tres subrutines que tenen la mateixa funció que les subrutines del gestor de memòria: *IDP_INIT*, *IDP_GET_ID* i *IDP_RELEASE_ID*.

El gestor multitasca (*taskmanager.X68*) conté *TM_INSTALL* que inicia la memòria dinàmica i els identificadors. Posa el temps mínim de cicle a zero. Prepara la llista de TCB (*Task Control Block*), consistent en el TEP (*Task Entry Point)*, el punter a la tasca, el TMP (*Task Memory Pointer*), el punter a la memòria de la tasca, i el TID (*Task ID)*, l'identificador de la tasca. Instal·la el *TRAP #0* i passa a mode usuari. El *TRAP #0* conté totes les funcionalitats encarregades de fer funcionar la multitasca cooperativa, amb totes les subrutines necessàries.

La segona part és la programació del joc (*tasks.X68)*, que es trobarà íntegre dins el document i que farà ús de les tasques. Sobre temàtica i més es té llibertat.

- dmm.X68

El gestor de memòria dinàmica s'encarrega d'administrar l'espai de memòria disponible per a les tasques.

A cada tasca se li assigna un espai de memòria de 200 bytes, del qual pot disposar per guardar-hi dades, paràmetres, valors que usarà en el futur…

Dins *dmm.X68* trobam dues variables: *DMM_MEMORY*, un espai de memòria suficient per guardar-hi els 200 bytes de cada tasca. S'obté multiplicant el *TM_MAX_TASKS* per el *TM_TASK_MEMORY_SIZE*. I *DMM_MEMLIST*, una llista dels punters a cada tros de memòria.

A *dmm.X68* trobam tres subrutines principals:

DMM_INIT, que s'encarrega de crear la llista *DMM_MEMLIST* i d'omplir-la amb els punters a cada espai de memòria. Aquesta subrutina ja estava escrita.

DMM_ALLOCATE s'encarrega de donar un tros de memòria sol·licitat. Cerca el primer espai disponible i retorna el punter a aquest espai. Després fa ús de la subrutina *DMM_CLEAR_AREA*, que tan sols s'encarrega de posar tots els bytes del tros de memòria a $00 mitjançant un senzill bucle. Ambdues subrutines ja estaven escrites.

DMM_RELEASE s'encarrega d'alliberar l'espai de memòria passat per paràmetre. Alliberar-lo significa tornar-lo a fer disponible. El procés consisteix tan sols en passar el punter donat i la direcció de *DMM_MEMLIST* i tornar a posar el punter dins la llista, per tal de fer-lo disponible. La instància anterior del mateix punter serà ocupada amb aquest mateix o amb altres *DMM_RELEASE*.

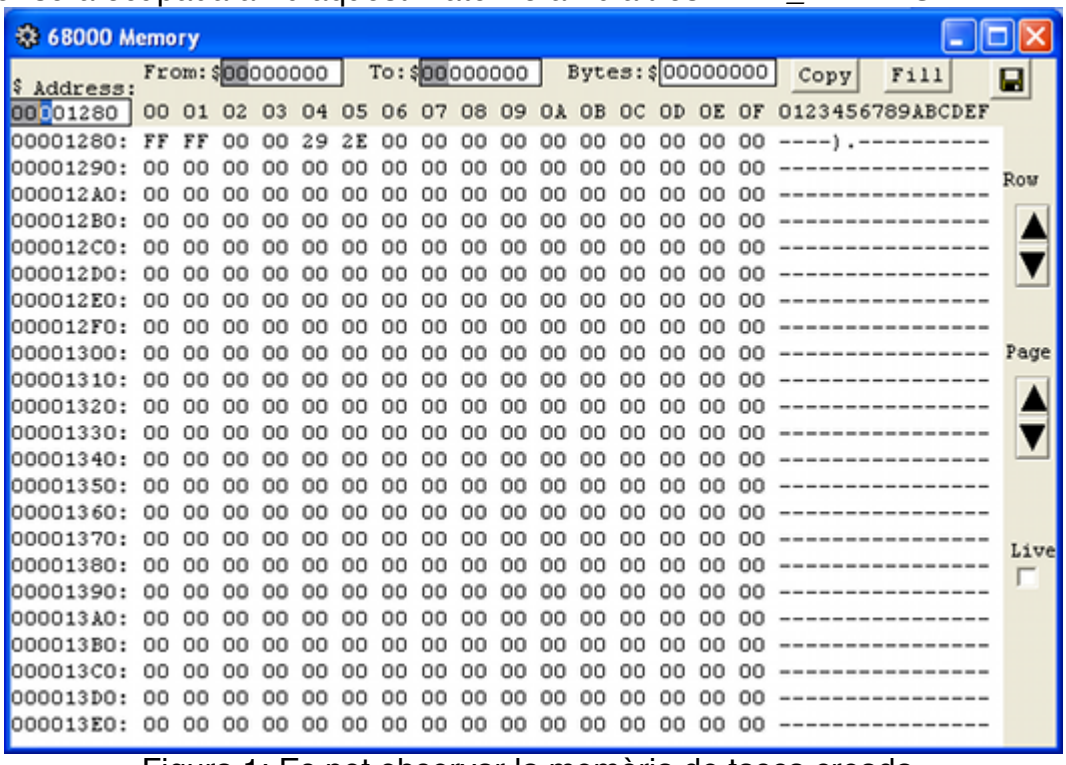

Figura 1: Es pot observar la memòria de tasca creada.

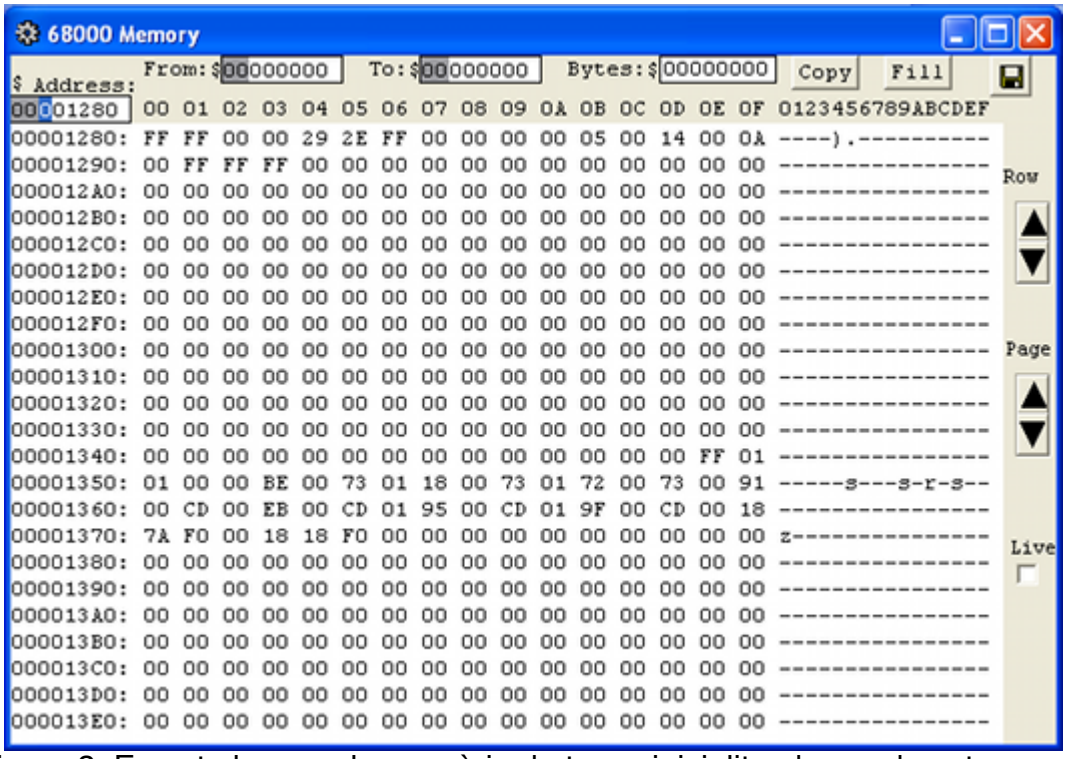

Figura 2: Es pot observar la memòria de tasca inicialitzada per dues tasques.

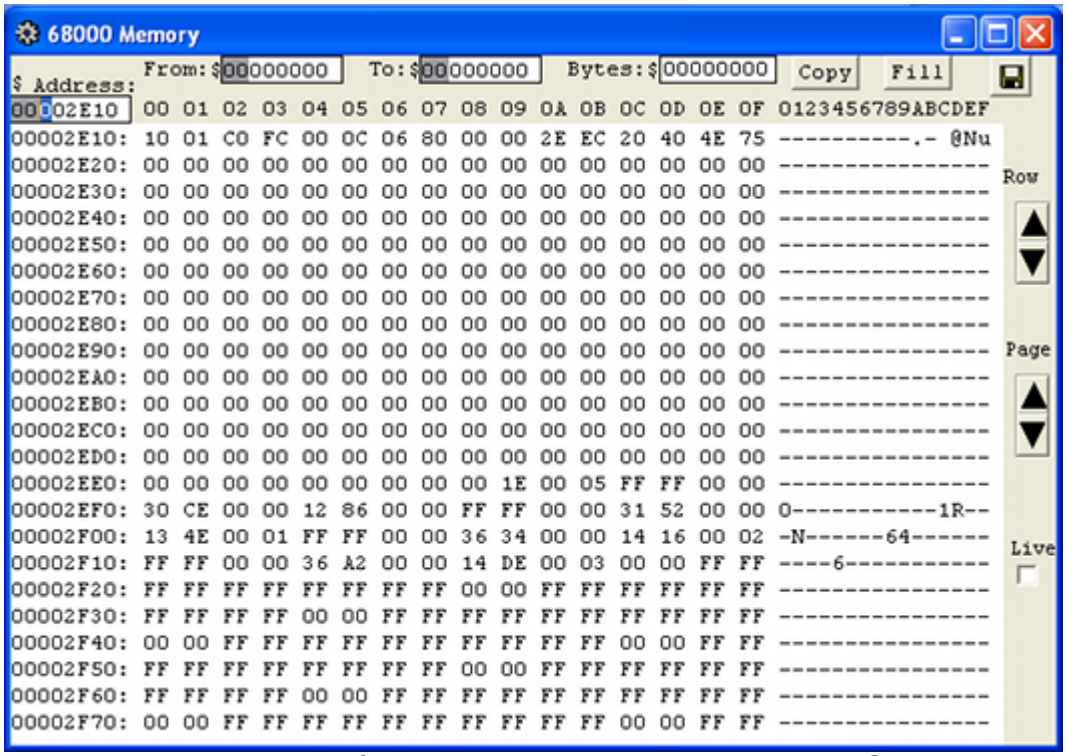

Figura 3: Memòria compartida, sobre la llista de TCB.

- idpool.X68

La llista d'identificadors conté tots els identificadors disponibles per a les tasques. Cada cop que s'entri una nova tasca dins el gestor multitasca, se li

assignarà un identificador que és únic entre les altres tasques, encara que siguin del mateix tipus.

Dins *idpool.X68* trobam tres subrutines:

IDP_INIT s'encarrega de crear la llista *IDP_IDLIST* i d'omplir-la amb els identificadors, que simplement seran nombres successors des de 0 fins a *TM_MAX_TASKS.* Aquesta subrutina ja estava escrita.

IDP_GET_ID s'encarrega de proporcionar un identificador disponible. Amb la llista d'identificador, cerca el primer element disponible i el retorna. A més, després l'elimina de la llista per indicar que ja no està disponible.

IDP_RELEASE_ID allibera l'identificador passat per paràmetre. Simplement transforma l'identificador a word i el torna a col·locar dins la llista per fer-lo disponible de nou.

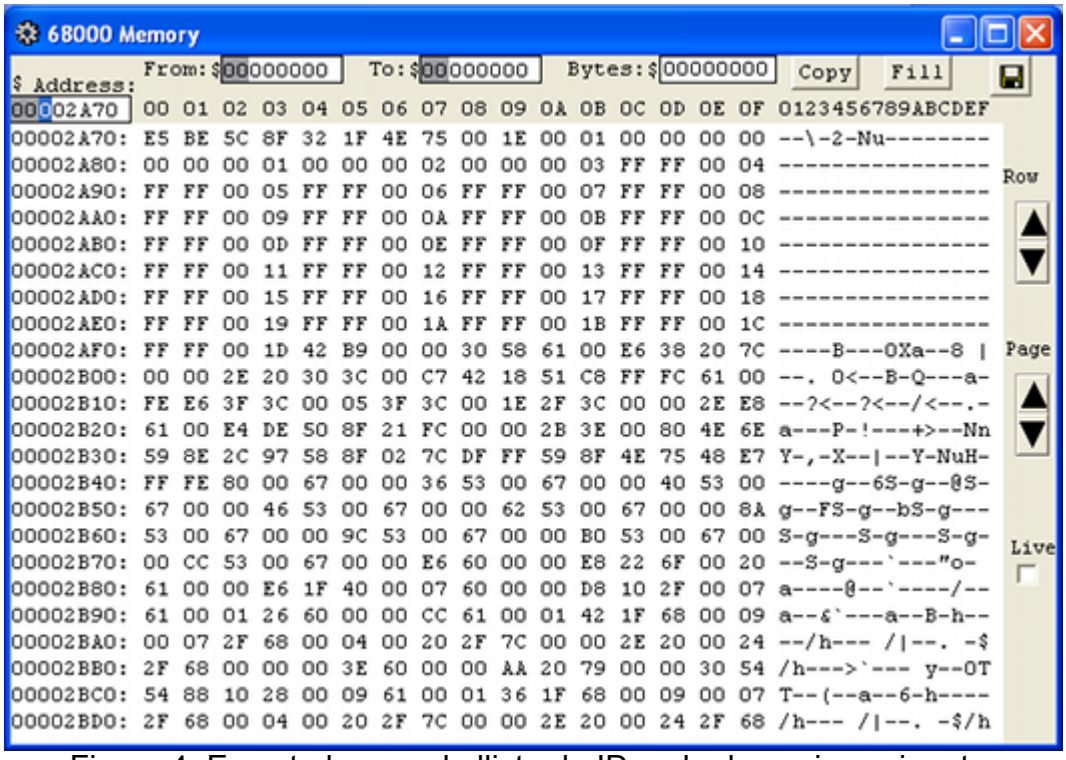

Figura 4: Es pot observar la llista de ID amb alguns ja assignats.

- taskmanager.X68

El gestor multitasca du a terme la programació cooperativa, que consisteix en l'execució de cada tasca una darrera de l'altra en un bucle infinit. Per a fer-ho usa les funcionalitats del *TRAP #0* especificades en l'enunciat proporcionat.

Dins *taskmanager.X68* està la llista de TCB (*Task Control Block)* formada per, en ordre, el punter a la tasca (*Task Entry Point, TEP)*, el punter a la memòria de la tasca (*Task Memory Pointer, TMP)* i l'identificador de la tasca (*Task ID, TID).* A més, també hi ha la memòria compartida, un espai de memòria accessible per a totes les tasques.

Trobam el *TM_INSTALL*, que s'encarrega de iniciar tots els processos (gestor de memòria dinàmica, llista d'identificadors, llista de TCB, temps mínim de cicle…). Després, com en el *TRAP #15,* mitjançant D0 s'elegeix quina funció del *TRAP #0* s'executa.

Les funcionalitats del *TRAP #0* són vuit i les que es troben ja escrites són: la 0, afegeix una nova tasca; la 1, elimina una tasca; la 4, obté informació de la tasca actual; la 5, obté informació a través de l'identificador, i la 8, estableix el mínim temps de cicle.

La funcionalitat 2 inicia l'execució del gestor. Guarda el temps d'inici, obté el primer element de la llista de TCB i guarda el punter a *TM_CURRENT_TASK.* Després guarda l'identificador, el punter a la memòria de la tasca i el punter a la memòria compartida en els registres adequats dins la pila, i per finalitzar, modifica el PC dins la pila per saltar directament a la tasca en retornar de la interrupció.

La funcionalitat 3 executa la següent tasca dins la llista. Obté la tasca actual mitjançant *TM_CURRENT_TASK,* de la qual obté el seu identificador, que usarà per cercar la tasca dins la llista de TCB amb la subrutina *TM_SEARCH_BY_ID.* Amb l'element, cercarà la següent tasca dins la llista. Si resulta que era l'última tasca de la llista, calcularà el temps d'execució del cicle fent ús de la funcionalitat 8 del *TRAP #15* i el compararà amb el *TM_PREVIOUS_TIME.* Si la diferència és menor que el temps mínim de cicle, farà un retràs per tal de suplir la diferència. Seguidament, obtendria el nou temps d'inici de cicle i tornaria al principi de la llista. Ja amb el següent element, faria igual que la funcionalitat 2, obtenint les dades necessàries i modificant el PC.

La funcionalitat 6 retorna informació de la primera tasca de la llista. Cerca el primer element de la llista de TCB i obté el *Task ID* i el punter a la memòria de la tasca. A més, obté l'índex relatiu de la tasca, que al ser el primer es tracta de 0. Amb relatiu ens referim a la seva posició dins les tasques que hi ha, no sobre la seva posició en la llista. Aquest índex ens serveix per fer una iteració de totes les tasques de la llista.

La funcionalitat 7 retorna informació de la següent tasca respecte a l'índex passat per paràmetre. Per trobar l'element iteram dins la llista. Com l'índex és relatiu a les tasques dins la llista, iteram tantes vegades com indica l'índex. Amb l'element ja trobat, obtenim el *Task ID* i el punter a la memòria de la tasca. L'índex a retornar és un posterior a l'actual, per tant, tan sols incrementam en un l'índex passat per paràmetre.

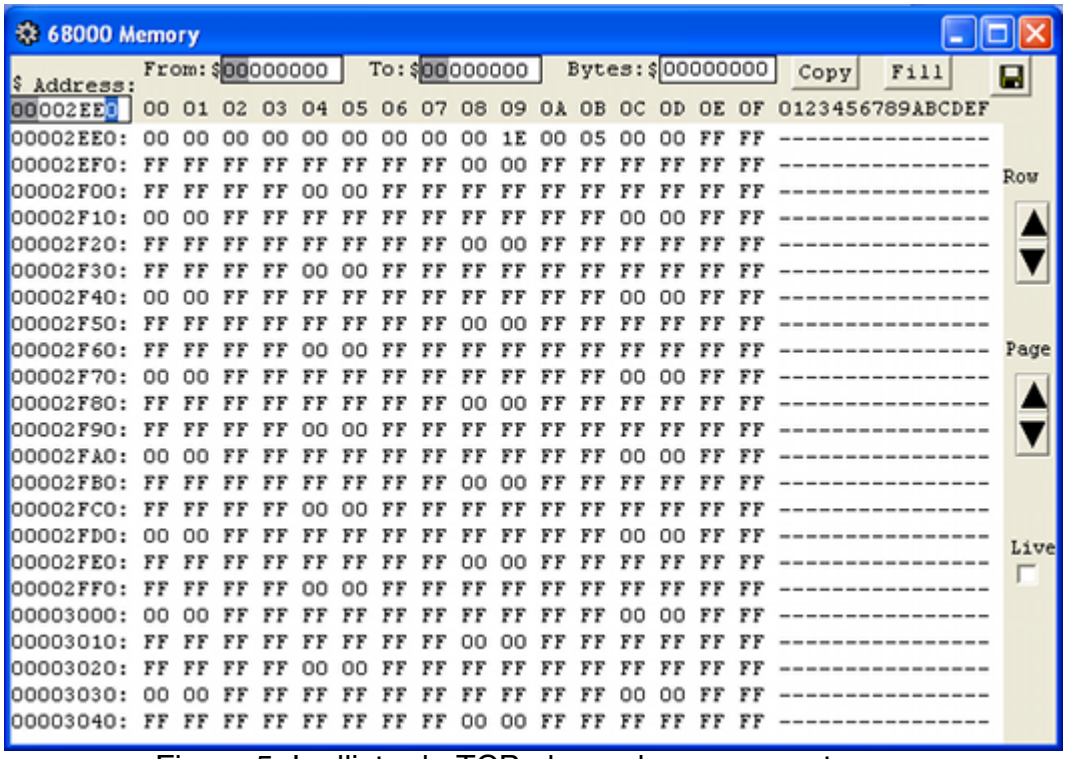

Figura 5: La llista de TCB abans de posar cap tasca.

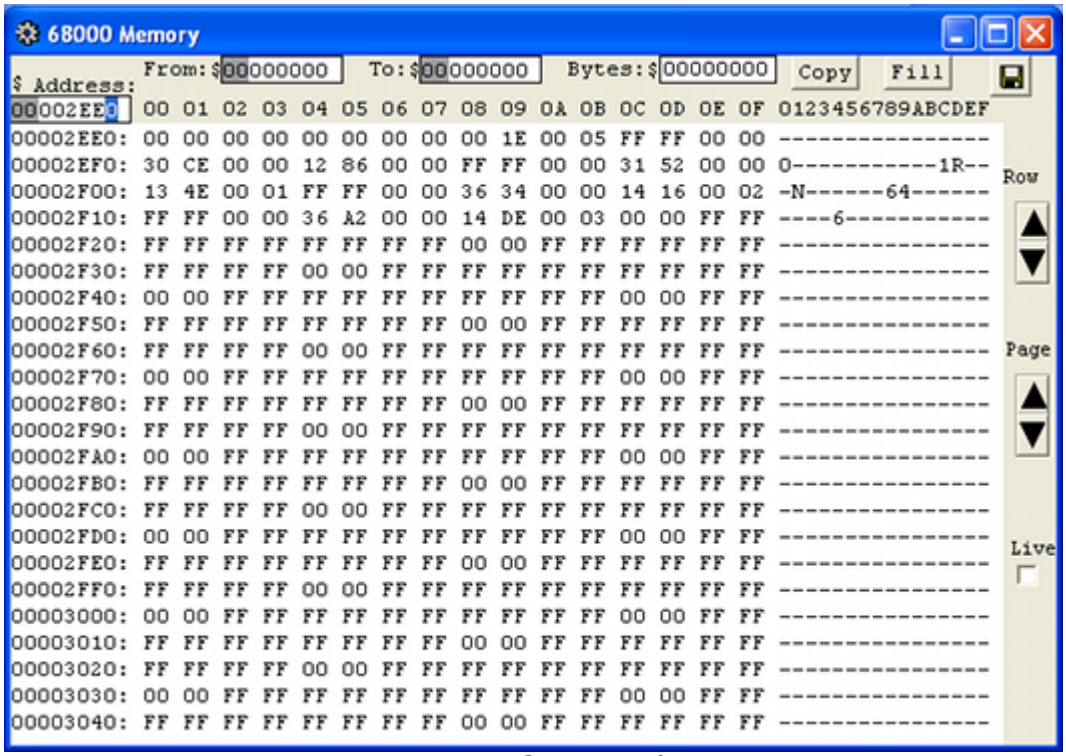

Figura 6: La llista de TCB després de TK_INIT.

- tasks.X68

El nostre joc, que hem anomenat JET TASK, té catorze tasques en la seva primera fase y un seguit de subrutines auxiliars que empleen distintes tasques.

Task 1:
És la tasca encarregada de dibuixar l'espai estrellat que es pot veure durant tota l'execució del joc, per tant es una tasca que mai s'eliminara mentre duri l'execució. Es dibuixen les estrelles (punts blancs) amb una separació constant entre elles però variant la seva posició al començament de cada línia depenent de la línia a la que es troben.

Task 2:
Aquesta tasca es l'encarregada de dibuixar el nom del joc. Això es fa mitjançant línies de color i després, un cop completada la lletra, es pinta l'interior per poder continuar amb la següent lletra en cas de que s'hi hagi d'haver una altra, quan ha pintat tot el nom també escriu, aquesta vegada mitjançant cadenes de text, "Press ENTER to continue" a la base de la pantalla per passar a la tasca 3

Task 3:
La tercera tasca escriu la historia introductòria del joc mitjançant cadenes de text.
Aquesta historia queda centrada al mig de la pantalla i igual que la tasca anterior torna a escriure "Press ENTER to continue" a la base de la pantalla per passar a l'execució del joc.

Task 4:
És la tasca que dibuixa GAME OVER enmig de la pantalla, així com les tasques anteriors apareixien al començament del joc aquesta apareix un cop s'ha eliminat la tasca que es refereix nau que es controla durant l'execució del joc, es a dir, un cop s'acaba el joc. Igual que a la segona tasca a aquesta també es dibuixa mitjançant línies i omplint cada lletra abans de començar a pintar l'altre i també mostra a la base de la pantalla "Press ENTER to continue" per reiniciar el joc.

Task 5:
Aquesta tasca s'executa quan es guanya la partida, de manera que mostra per pantalla CONGRATULATIONS seguit de una referencia a la historia inicial del joc. Igual que les tasques anteriors, també mostra a la base de la pantalla "Press ENTER to continue".

Task6:
Aquesta tasca simplement activa el doble buffer i mostra per pantalla l'execució del joc.

Task 7:
La sèptima tasca és la que detecta si s'ha pitjat colque tecla, per poder actuar després segons les tecles detectades. No les comprova totes ja que hi ha moltes tecles que no es fan servir, simplement comprova les fletxes amunt i avall

(encarregades del moviment de la nau), l'espai (per saber si la nau dispara) i l'enter (per canviar de tasca en cas de que sigui necessari).

Task 8:

Aquesta tasca està encarregada de dibuixar la nau.

En l'inici col·loca la nau en el centre de la pantalla, i inicia els seus colors.

En l'execució, posa els colors, obté el punt inicial operant amb la coordenada guardada en la memòria i el radi de la nau, i dibuixa les línies que conformen la nau. Després es situa a dins i l'omple del color seleccionat.

Task 9:

Aquesta tasca controla una nau enemiga, gestionant la seva posició, moviment i quan dispara.

En l'inici, posa les coordenades inicials. La coordenada X sempre és la mateixa, l'extrem de la finestra, però la coordenada Y l'elegeix amb una pseudo-aleatorietat. Després inicia la velocitat segons el tipus d'enemic de que es tracti, que la tasca ho pot saber amb el punter que ha inicialitzat el creador d'enemics (tasca 12). Per acabar, inicia els colors.

L'enemic canvia de color en cada cicle. Com el color és un nombre en hexa-decimal, en cada cicle l'increment en un, reiniciant quan arriba al màxim. Després el dibuix usant les coordenades inicials i la constant costat de l'enemic. Seguidament comprovam si l'enemic dispara. Per saber-ho, controlam el comptador de dispar de l'enemic: quan arriba a 0, el reinicia, crea una bala enemiga i posa el seu camp de disparat a afirmatiu. Després actualitzam les seves coordenades segons la seva velocitat. Si toca l'extrem superior o inferior de la finestra, rebota. Si en canvi toca l'extrem esquerre, eliminam la nau enemiga. Per finalitzar, comprovam que no toqui la nau protagonista, obtenint les seves coordenades, i comparant-les amb les pròpies. Si coincideixen, es crea la tasca de fi de joc.

Task 10:

Aquesta tasca controla la bala pròpia. La dibuixa i mira si ha donat a un enemic.

En l'inici, obtenim les seves coordenades inicials. La X és fixa, la Y l'ha guardada la nau protagonista en memòria compartida. Després inicialitzam la velocitat i el color.

En l'execució, posam el seu color i la dibuixam. Tenim una costant que és la longitud de la bala. Seguidament comprovam si ha ferit a un enemic, obtenint les seves coordenades. En cas afirmatiu, obtenim el seu identificador i l'eliminam. A més, posam l'identificador de la nau a l'*array* inicialitzat pel *bullet collector* (tasca 14), per posteriorment eliminar la bala. En cas negatiu, actualitzam les coordenades de la bala.

Task 11:

Aquesta tasca controla les bales de les naus enemigues. Una tasca representa tres bales. Administra el seu moviment i si ha ferit a la nau protagonista.

A l'inici cerca una nau enemiga que hagi disparat. Un cop trobada, obté les seves coordenades i les assigna a les tres bales. Després reinicia el camp de disparat de la nau enemiga i inicialitza la booleana que controla si cada bala s'ha eliminat o no.

En executar-se la tasca, per cada bala comprovam si la bala segueix en pantalla. Si es així, actualitzam les seves coordenades i la dibuixam. Seguidament miram si la bala ha sortit de la finestra. En cas afirmatiu ho marcam, per no tenir en compte la bala en el següent cicle. En cas negatiu, mitjançant una subrutina auxiliar miram si la bala ha ferit a la nau protagonista. Finalment, si les tres bales tenen el camp d'eliminat marcat, es posa l'identificador de la tasca a l'*array* inicialitzat pel *bullet collector*.

Task 12:

Aquesta tasca està encarregada de gestionar la creació d'enemics. Administra quants enemics crea i quan els crea.

En l'inici, mitjançat una pseudo-aleatorietat, elegeix quina seqüència d'enemics sortirà, per tal que no es repeteixi en cada partida. La tasca inicia un punter a l'*array* que té el nombre d'enemics que surt per tongada i un altre punter al tipus d'enemics. A més, inicia el comptador que decidirà quan es creen enemics.

En executar-se la tasca, decrementa el comptador, i si arriba a 0, el reinicia i mira quants enemics ha de crear. Si es troba un 0, significa que no ha de crear més enemics i per tant el jugador ha guanyat el joc. Si es troba un nombre distint a 0, l'usa per executar un bucle en el qual crearà un enemic en cada iteració.

Task 13:

Aquesta tasca està encarregada de representar i controlar la barra de vida de la nau.

En l'inici tan sols posa en una variable de vida el màxim nombre de tocs que pot rebre la nau.

En executar-se la tasca, mira si la variable booleana que controla que han ferit la nau està afirmativa. Si no és així, dibuixa directament la barra de vida. En cas afirmatiu, resta un a la vida de la nau i procedeix a dibuixar-la. Primer dibuixa el contorn de la barra de vida, i després en un bucle dibuixa cada rectangle representant la vida. En cas de ser 0, també crea la tasca de fi de joc.

En aquesta tasca una excepció por ocórrer. En el cas que en arribar la vida a 0 ja es trobin 30 tasques executant-se, no es podrà crear la tasca de fi de joc, per tant, no acabarà el joc i la barra de vida es disparará, a causa del seu codi.

Task 14:

Aquesta tasca, anomenada *bullet collector*, s'encarrega d'anar eliminant totes les bales que s'han sortit de de la finestra o que han tocat a una nau.

En l'inici, situa un punter en un *array* que es troba a memòria compartida, on les bales que es vagin eliminant posaran el seu identificador. A més, indica el nombre màxim de bales permeses per a la nau protagonista.

En executar-se la tasca, aquesta posa un 'fi d*array*' a memòria compartida, per saber quan aturar-se d'eliminar. Després procedeix a eliminar la tasca lligada a cada identificador que troba dins l'*array* fins el 'fi d'*array*'. A més, cada cop que elimina una tasca, mira si es correspon a una bala pròpia. En cas afirmatiu, incrementa el nombre de bales pròpies per tal de poder seguir disparant. Al final, posa el punter a l'inici de l'*array* de nou per repetir el procés.

Seguidament, trobam un exemple. En l'exemple (1) veim com la bala fereix a la nau i procedirem a posar $FF a D7, que en retornar de la subrutina a la que es

troba, la posarà en memòria (2). Aquest canvi s'observa en la Figura 8.

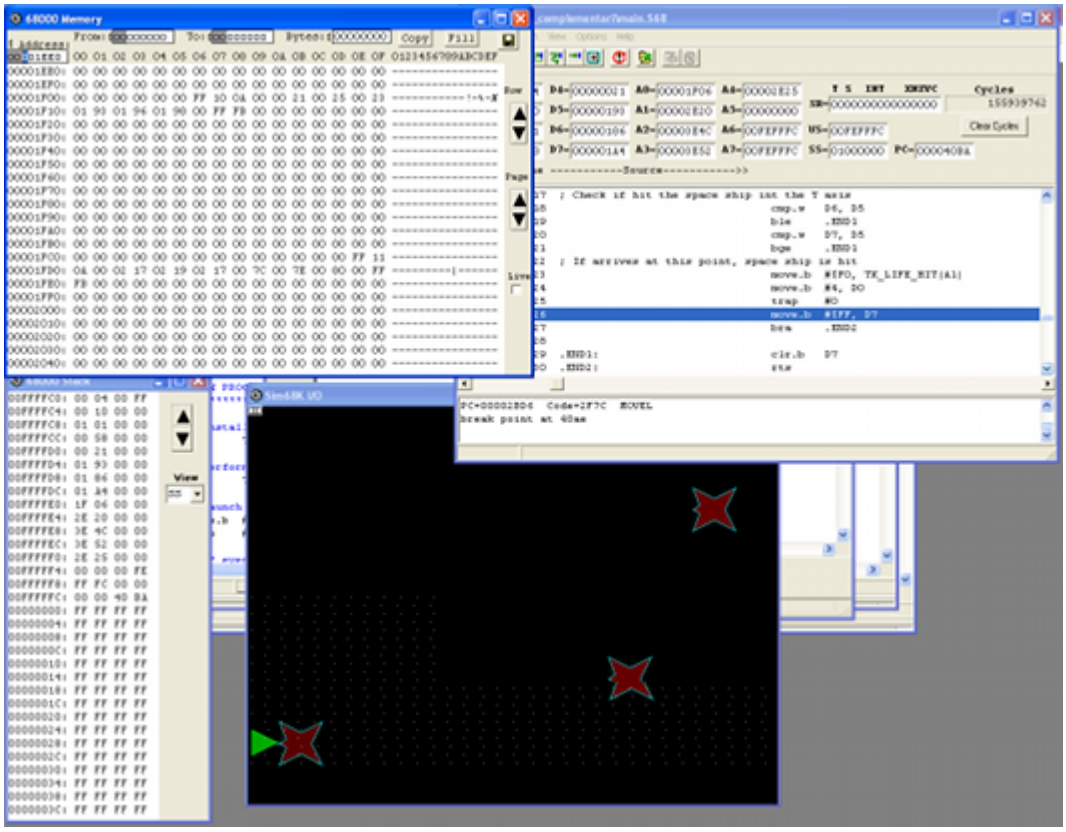

Figura 7: Exemple de la modificació de la memòria en el joc (1).

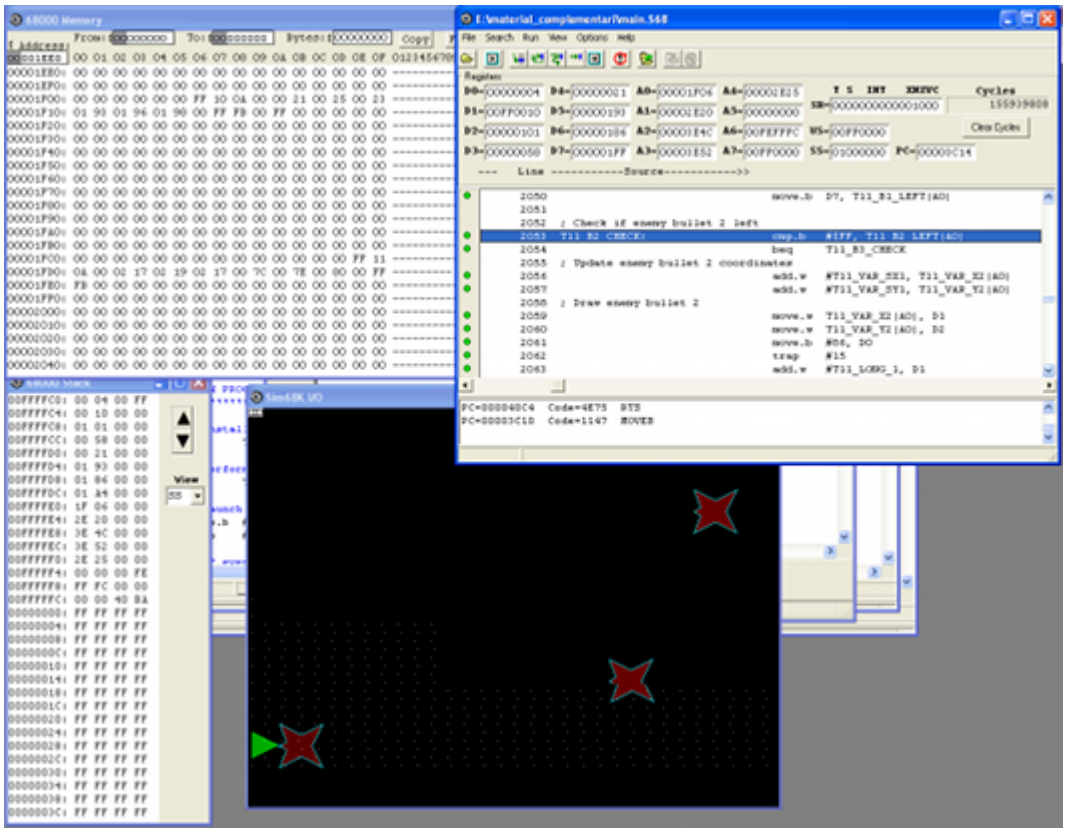

Figura 8: Exemple de la modificació de la memòria en el joc (2).