Emulador GBC: Aprendre i jugar en la seva màxima expressió

Escola Politècnica Superior d'Enginyeria de Vilanova i la Geltrú

Aleix Abengochea Molar

20 d'octubre de 2021

Resum

Les videoconsoles antigues són una font perfecta d'informació i coneixement. S'han originat comunitats enormes al seu voltant, donant pas a un ecosistema que permet l'estudi de diferents camps de la computació amb un objectiu definit i sobretot interessant.

L'objectiu d'aquest treball té com a fita la implementació d'un emulador de la videoconsola Gameboy de Nintendo. Primer es realitza un breu resum sobre el funcionament dels principals components que conté. Tot seguit s'explica quina ha estat la planificació, el disseny i la implementació de l'emulador. Després es comenten els diferents problemes que s'han anat afrontant a mesura que el treball anava progressant i finalment es plantegen diverses funcionalitats o millores que no han estat contemplades en aquest treball per manca de temps.

En aquest projecte, ens endinsem de ple en entendre el funcionament dels mecanismes interns d'un computador simple. Com tractar la programació a baix nivell des d'un llenguatge de programació modern i com dissenyar una aplicació que porti a la pràctica tots aquests conceptes.

1 Introducció

Aprendre a multiplicar números sense saber sumar pot ser útil en molts aspectes, però mai et portarà a plantejar-te certes coses o seràs incapaç de donar resposta a moltes preguntes. Crec amb molta convicció, però sense gaires proves que Entendre la part més baixa de la piràmide que estudiem és fonamental per donar un fil lògic i explicació a tot el que creem.

Els computadors com a tal s'han complicat de manera exponencial amb el pas del temps. Tenen múltiples etapes, diferents nivells de memòria cau i fins i tot predictors de salt.

A mesura que evoluciona la tecnologia, tenir una idea general del funcionament de les coses és cada vegada més complicat. Per aquest motiu, penso que estudiar i recrear el comportament físic de la Gameboy es troba en el punt d'equilibri perfecte entre l'enteniment i la complexitat de molts conceptes.

La Gameboy ofereix un escenari de partida perfecte per tal d'assentar una base de com funcionen els microprocessadors en general. Té molt bona documentació, una comunitat sòlida, i fins i tot més versions que poden anar complicant i complementat les primeres.

En l'àmbit personal, un dels camps de la informàtica que sempre m'ha interessat és la part lògica i programació a baix nivell. Quan vaig ser alumne de la Universitat Rovira i Virgili, vaig aprendre ARM a través de la Ninstendo Ds (NDs). Això va fer créixer la meva motivació, ja que era el primer cop que veia una aplicació molt pràctica i real d'un àmbit que m'agradava.

A part de tot això, la Gameboy probablement va ser el primer component electrònic que vaig tenir a les mans i un dels que segurament li he dedicat més hores. Guardo bons records i d'alguna manera aquí es tanca el cicle.

2 Gestió del projecte

El desenvolupament d'un software com el que he creat està condicionat notablement per la dificultat de comprensió inicial que suposa. Hi ha molt bona documentació però els conceptes al principi són feixucs d'entendre. Això provoca que fer una estimació inicial i real del projecte no sigui una tasca senzilla.

Per tal de tenir una idea i un fil conductor del projecte, s'ha portat a terme una primera planificació inicial basada en el nombre d'hores que normalment és dediquen a fer el TFG i a una primera estimació molt bàsica de quina feina comportaria el projecte.

Aquesta estimació parteix de les 450 hores totals que s'acaben dividint de la següent forma:

- **Llegir/Planificació:** 30 hores.
- **Mapa conceptual:** 30 hores.
- **Modelat en classes:** 30 hores.
- **Implementació:** 180 hores.
- **Afegir Swing:** 30 hores.
- **Joc de proves:** 30 hores.
- **Redacció:** 60 hores.
- **Revisió/Imprevistos:** 60 hores.

A mesura que el projecte va anar avançant es van produir molts canvis. El fet d'entendre millor l'abast del projecte va anar modificant enormement les fites programades. No hi ha una planificació final acurada perquè moltes de les tasques s'havien de portar en paral·lel o es desenvolupaven en situacions diferents.

S'ha realitzat una planificació final per tal de poder comparar el temps inicial amb el final la qual ha suposat un increment d'hores fins arribar a les 540. Puc dir que les tasques que més feina extra han portat han estat tant la documentació com la implementació.

Finalment, pel que fa a la gestió del projecte, s'ha de parlar sobre l'avaluació econòmica. En aquest punt es va decidir plantejar un escenari basat en un treballador autònom que oferia el projecte com un producte o servei a una empresa o agent extern.

Per tal de simular un entorn real, s'han definit diferents costos: personal, maquinari, programari, generals i contingència. Cadascun basats en com està el mercat actualment. També és fa una diferencia entre quin és el pressupost inicial abans de començar el projecte i el real.

Al final s'obtenen aquest dos resultats:

- **Pressupost inicial:** 14564 euros.
- **Cost total:** 15197 euros.

La diferencia entre el pressupost inicial i el cost total no acaba sent gaire gran gràcies al cost de contingència que es planteja inicialment.

3 Especificacions tècniques

La Gameboy és una videoconsola llençada al mercat l'any 1989 per la companyia Nintendo [1]. No va ser la videoconsola més innovadora de l'època ni la que tenia les millors prestacions, però en general, va acabar sent un èxit mundial [2]. Va tenir diverses versions i millores, però aquest projecte es centra en la primera, també coneguda com a DMG.

En línies generals, les especificacions, expliquen basats en la documentació i experiències propies quin és el funcionament dels components interns de la Gameboy.

Els elements que s'expliquen són tots els que es poden observar a la figura 1 a excepció del sistema de so i la connexió serie. A més a més també es defineixen com són els cartutxos de la videoconsola ja que també tenen components i lògica interna.

4 Disseny i implementació

El disseny i la implementació ha estat el gruix del projecte. S'ha seguit una metodològica on s'intentava que cada part funcionés per separat encara que això signifiques fer múltiples iteracions del projecte. El resultat i com es mostra aquí, només contempla la manera en com es portaria a terme un cop vist des de fora. L'ordre real dels elements ha estat més caòtic ja que molts parts necessiten assaig i error per tal d'arribar a comprendre realment el seu funcionament.

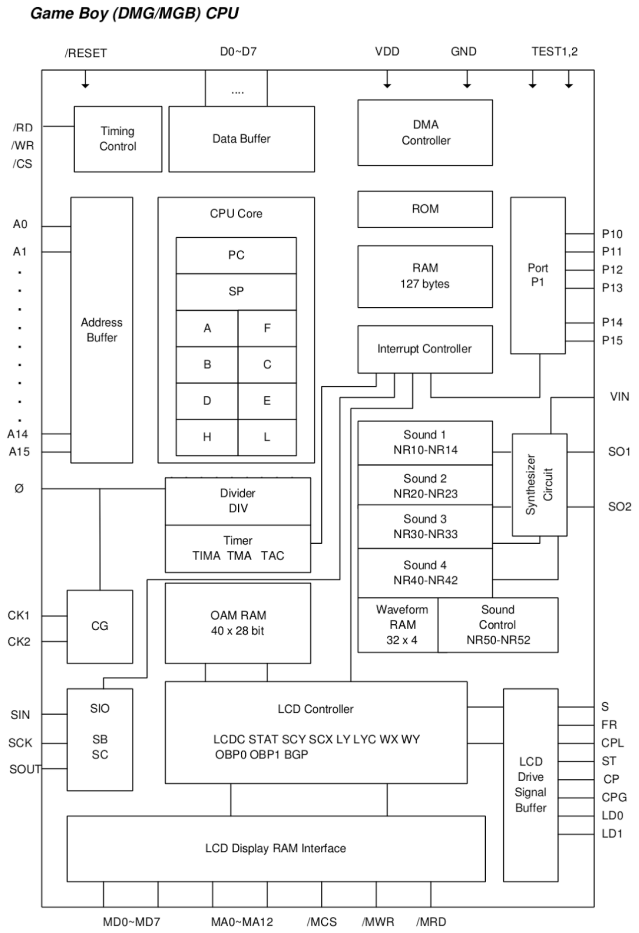

Figura 1: Esquema intern [3]

La implementació d'aquest projecte s'ha portat a treme utilitzant Java i fent ús del paradigma orientat a objectes.

Per tal de poder tenir un emulador funcional, s'ha implementat i establert les següents etapes:

Sistema bàsic

Petit sistema inicial que serveix com a prova de concepte. És un prototip de CPU totalment funcional el qual té molt poques operacions implementades.

Memòria

Memòria o registres, es un component que abstreu el concepte de memòria per tal de poder ser implementat d'una manera més eficaç i no utilitzant una simple matriu la qual més endavant és difícil de gestionar.

Mmu

A la Gameboy, tots els components es comuniquen utilitzant adreces de memòria. Aquesta etapa, defineix un sistema de com gestionar el bus d'adreces i permetre la comunicació entre els diferents elements.

Cart

Defineix com són els cartutxos bàsics de la Gameboy que contenen la informació d'un joc.

Interrupcions

Sistema que permet interrompre l'execució de la CPU per tal de tenir control sobre els elements externs.

CPU

Component principal del emulador que gestiona totes les diferents operacions.

Display

Interficie on es representen els píxels en quatre colors. També inclou el PPU que és l'encarregat de formar la informació que s'enviarà al display.

Input

Defineix l'interficie amb la qual l'usuari es capaç de controlar el joc.

Clock

Element simple però clau del disseny que sincronitza tots els elements.

MBC

Reformula la definició prèvia de Cart per tal de acceptar més versions.

Interficie d'usuari

Implementació amb la llibreria Swing [4] de la interficie d'usuari.

Per acabar, un disseny esquematitzat de com ha quedat la implementació final es pot veure a la figura 2.

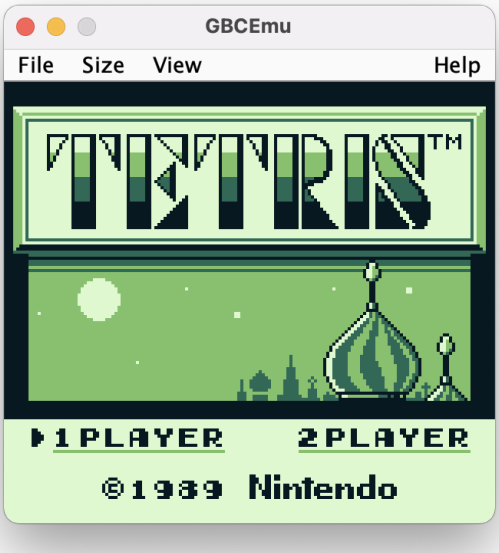

Figura 3: Aplicació final executant un joc

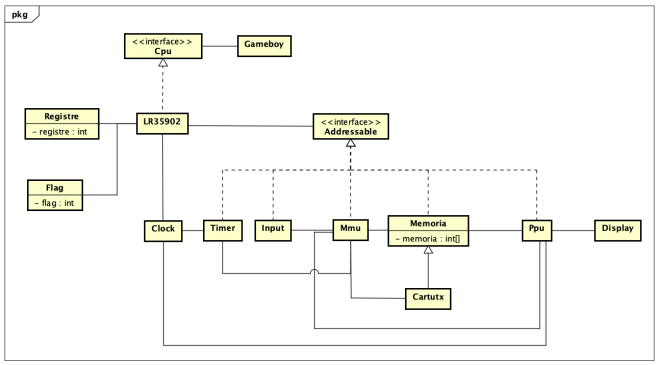

Figura 2: Diagrama final

I una petita mostra de com ha quedat l'aplicació final es pot veure a la figura 3.

5 Problemes

Ja que tant la teoria com la implementació de l'emulador és un tema feixuc i extens, trobo que és encertat dedicar un apartat a explicar els diferents problemes que s'han anat trobant durant el desenvolupament. És una manera de fer més entretinguda l'explicació i probablement l'apartat que serà més útil a aquelles persones que estiguin interessades a intentar afrontar el mateix repte.

En aquest apartat s'explicarà amb detall els problemes més importants que s'han anat afrontant a mesura que s'ha desenvolupat el treball. Estarà centrat en la implementació.

Tipus primitius en Java

El problema que més modificacions ha provocat és el fet d'haver triat Java com a llenguatge per a la implementació.

Per tal de ser rigorosos amb la implementació, en un principi, es van triar diferents tipus primitius de variable per tal que tinguessin la mateixa mida que la seva analogia real. Per exemple:

Per a totes les adreces de memòria i opcodes de la CPU, s'utilitzava el tipus short de Java perquè té una capacitat de 16 bits.

Per a totes les configuracions de memòria i registres, s'utilitzava el byte, ja que té una capacitat de 8 bits.

El problema ve donat quan intentes operar amb aquest tipus en Java. Java no realitza les operacions amb el tipus que hagis definit, sinó que fa una conversió interna al tipus int. Això genera diferents problemes:

El primer i més important, és que realitza una extensió de signe per defecte. Un valor que tingui el bit de més pes a 1 com el 0xFFFF, quan es transforma a un int passa a ser el valor 0x8000FFFF. I no hi ha cap manera de saltar aquesta funció i no existeixen els primitius sense signe. La solució que es va fer al principi és intentar fer comprovacions per cada operació que es portava a terme, però al final es va decidir tornar a implementar tot el projecte utilitzant int com el tipus per defecte. Això és possible, ja que cap dada omple els 32 bits i mai es té el problema del signe.

L'altre problema era que després de fer una operació, el resultat havia de ser guardat en una variable int o realitzar un casting al tipus en qüestió. Ja que en Java una operació entre dos bytes genera un int. La solució inicial va ser fer casting després de cada operació, però al final es va resoldre aquest problema quan es va tornar a implementar amb ints.

Rellotge de 8Mhz

Per tal de definir el temps dins el nostre emulador, es va intentar definir una classe clock que portés el temps dins el programa i organitzés els

altres components. Per exemple, aquesta classe clock tindria una funció run, que s'encarregaria d'activar la funció tick o step a tots els altres elements. Així aconseguiríem que tots els elements tinguessin un sincronisme entre ells. Aquesta idea d'implementació es basa en com funciona el hardware.

El problema d'aquesta idea inicial és una limitació del llenguatge i segurament de hardware. El processador de la Gameboy té una velocitat de rellotge de 8 Mhz, això vol dir que hem de ser capaços de fer tick cada 125 nanosegons.

És impossible fer una implementació que ens deixi executar una funció cada 125 nano segons. Hi ha una funció en Java que ens retorna el temps en aquesta magnitud, però a la documentació ja indica que té una imprecisió molt gran. Per altra banda hi ha massa overhead si intentem fer el càlcul de 125 ns i intentem executar tots els elements.

La solució a aquesta implementació va estar donar el control a la CPU. Executar una instrucció, portar un recompte de quants cicles teòrics comporta això i executar el mateix nombre de cicles a la resta de components.

Refactor Cartutx

Aquest va ser un refactor necessari perquè el codi es pogués entendre. La primera implementació recarregava la classe Cart de tal manera que qualsevol lectura o escriptura havia de passar per moltes comprovacions per tal d'identificar quin tipus de MBC havia d'utilitzar.

La solució ha estat dividir en múltiples classes per tal que futures implementacions de diferents tipus de cartutx no impliqui modificar totes les classes un altre cop. Ara en carregar, s'identifica el tipus i només es carrega l'algoritme necessari.

Display Jswing

Més que un problema ha estat un falta de comprensió de com funcionen els gràfics o el Swing a Java. Probablement si algú es troba en aquest problema, el millor consell és que utilitzi una llibreria o un framework que no sigui enfocat a finestres o entendre com funciona el EDG de Swing.

La llibreria Swing de Java genera un thread principal que serà l'encarregat de gestionar tots els elements gràfics del programa. El problema sorgeix quan intentem executar el nostre emulador i a la vegada tot el sistema de finestres sense utilitzar threads nous.

El display és una imatge que es refresca cada 16.7 mil·lisegons. El sistema no és capaç d'executar tot a la vegada. Estem obligats a generar un thread per a la nostra lògica de l'emulador i un per al funcionament principal de l'aplicació.

Debugger

La implementació de la CPU és molt propensa a errors. És important tenir alguna espècie de debugger o test que ens permeti saber que està passant a l'emulador en cada moment.

En aquesta implementació a causa de la falta de temps no s'ha inclòs una explicació detallada d'aquest comportament. Al codi font del programa s'adjunten diferents classes que es van utilitzar per a mostrar el comportament dels diferents components.

6 Conclusions

Encara que no s'han pogut assolir tots el objectius que s'havien plantejat inicialment, el resultat del treball és totalment satisfactori. No s'ha aconseguit implementar el funcionament de la Gameboy Color ni de les opcions de debugger, però el simple fet de veure executar el Tetris per primer cop fa que tot l'esforç hagi valgut la pena.

Per tot el que resta puc dir que fer aquest projecte t'ensenya moltíssim. Des de com lidiar amb operacions una mica més baix nivell fins a abstraccions enrevessades. Estic orgullós de poder-me definir com una persona que té per afició el mateix que ha estudiat. El fet d'haver fet aquest projecte em proporciona un escenari ideal per fer mil proves i implementacions esbojarrades que queden massa lluny de la línia general del projecte per a ser incloses en implementacions futures. Un altre aspecte important és com és d'interessant el projecte. En l'àmbit personal m'ha motivat molt i sé que qualsevol altre projecte m'hagués cansat o avorrit molt abans.

Recomano a qualsevol persona interessada una mica amb el tema al fet que s'animi a intentar-ho. Hi ha moltíssimes variants, molt bona documentació i ofereix moltes idees noves que es poden aplicar.

7 Implementacions futures

Moltes característiques han quedat fora de l'abast d'aquest projecte. Algunes per complexitat i moltes per la falta de temps.

En aquest apartat es defineixen els objectius principals que es voldrien aconseguir en una segona iteració d'aquest projecte o perquè altres persones interessades en fer un projecte no igual però similar les puguin adoptar com idees.

Més configuracions de MBC

Hi ha jocs que no funcionen en aquest emulador perquè no estan implementats tots els tipus de cartutx que existeixen. Només estan implementats els principals i més senzills. Una de les implementacions que es vol aconseguir tenir en una futura versió és tot l'espectre de MBC fins al 5. El qual comprèn jocs com el Pokémon.

La implementació no és gaire complicada, però té elements nous com guardar el joc de manera persistent en una memòria RAM alimentada per bateria o un rellotge real.

Sistema de so

No tenir sistema de so va ser una de les primeres decisions que es van prendre a l'hora de dur a terme aquest projecte. Era factible no implementar res del sistema de so, ja que no interactua amb cap altre component de l'emulador.

En una segona iteració d'aquest projecte, seria un dels principals objectius a assolir, ja que és una part fonamental per a l'experiència d'usuari.

Cpu algorítmica

Des d'un principi es valorava la possibilitat que les diferents instruccions de la CPU es generessin de manera automàtica. Utilitzant un algoritme que seguís la lògica real que té el disseny del hardware.

No va ser fins que ja estava molt avançat el projecte que vaig trobar la informació de com es podia portar a terme. Aquesta implementació no afectarà l'usuari final, però simplificarà moltíssim la classe CPU i permetrà entendre de manera més senzilla quin és el seu funcionament.

Debugger i visió interna

Un dels objectius principals del projecte era la possibilitat que la gent aprengués i entengués com funcionen els diferents components. En un principi es volia tenir una opció per desenvolupadors que presentés d'una manera senzilla la informació de les diferents memòries, dels registres i de com es formen els gràfics.

Es marca com a objectiu el fet de crear una interfície per a tots els components interns per tal de visualitzar-los i modificar-los en temps d'execució.

Serial i connexió remota

Una de les connexions de la Gameboy és una connexió sèrie que permet connectar diferents videoconsoles entre elles. Això permet jugar o compartir dades en funció del joc.

En una segona versió del projecte es vol implementar aquesta lògica per tal d'intentar connectar dues instàncies de l'emulador en remot.

També seria interessant intentar crear un adaptador per tal de connectar una Gameboy física amb una emulada.

Gameboy Color

En un principi, aquest treball contemplava la implementació d'un emulador de la Gameboy Color. Al final només s'ha emulat la seva versió base, però és un dels objectius per a futures versions.

Guardar l'estat del emulador

Guardar l'estat del joc actual és una de les opcions que tenen gairebé tots els emuladors. En aquest treball no es va contemplar aquesta funció des d'un principi i la implementació requereix temps. S'han de serialitzar tots els objectes que guarden informació (Ram i registres) i guardar-los en un fitxer.

Per a una pròxima versió seria una bona funcionalitat a tenir en compte.

Iniciar un altre joc

La manera d'implementar alguns components utilitzant Singletons ha provocat que reiniciar tots els components no sigui trivial. Per la qual cosa la versió actual de l'emulador no permet iniciar un segon joc.

Es preveu reformular aquests components o implementar alguna funció reset per tal de resoldre el problema.

Android

L'única cosa bona que podem treure d'haver utilitzat Java com a llenguatge és que podem fer la portabilitat a Android amb relativa facilitat. Únicament haurem de reformular l'input i la interfície d'usuari, ja que tota la part lògica hauria de ser equivalent.

En futures versions es contemplaria l'opció de generar una apk.

8 Agraïments

En primer lloc m'agradaria agrair al professor Bernardino Casas per oferir-me l'oportunitat de fer aquest treball. Sense la seva dedicació i atenció aquest treball no hauria estat possible.

També vull agrair a la meva parella pel fet de recolzar-me i motivar-me en tot moment, als meus familiars per no perdre l'esperança i al meu pare al qual li hagués encantat el resultat.

Per últim incloure a tota la comunitat que recull i comparteix informació de manera altruista i fa que projectes com aquest siguin possibles.

Referències

[1] author. "Game Boy." (), adr.: https://es.wikipedia.org/wiki/Game_Boy.

[2] L. S. Vailshery. "Video game console sales worldwide for products total lifespan as of September 2021." (jul. de 2021), adr.: https://www.statista.com/statistics/268966/total-number-of-game-consoles-sold-worldwide-by-console-type/.

[3] author. "DMG." (), adr.: http://dot-matrix-game.blogspot.com/2014/01/.

[4] ——, "Swing." (), adr.: https://docs.oracle.com/.

[5] A. N. Díaz, A. Vivace, Beannaich, E. ". H. Cory Sandlin, Elizafox, Furrtek, Gekkio, J. Frohwein, J. Harrison, L. ". Halphon, Mantidactyle, M. Fayzullin, M. ". Korth, P. of ATX, P. Felber, P. Robson, T4g1, TechFalcon, endrift, exezin, jrra, kOOPa, mattcurrie, nitro2k01, pinobatch, P. Fagan i A. Burnett. "Pan Docs." (), adr.: https://gbdev.io/pandocs/.

[6] J. Frohwein. "DMG Schematics." (), adr.: https://gbdev.gg8.se/wiki/articles/DMG_Schematics.