Universitat Autònoma
de Barcelona

LittleProc 2.0. L'Evolució als 16 bits, amb arquitectura superescalar i amb pipeline

Memòria del projecte
d'Enginyeria Tècnica en
Informàtica de Sistemes
realitzat per
Roc Prat López
i dirigit per
Raül Aragonès Ortiz

Escola d'Enginyeria
Sabadell, *Juny* de *2011*

[El/La] sotasignat, ***Raül Aragonès Ortiz***, professor[/a] de l'Escola d'Enginyeria de la UAB,

CERTIFICA:

Que el treball al que correspon la present memòria ha estat realitzat sota la seva direcció per ***Roc Prat López***

I per a que consti firma la present. Sabadell, ***Juny*** de ***2011***

Signat: ***Raül Aragonès Ortiz***

FULL DE RESUM – PROJECTE FI DE CARRERA DE L'ESCOLA D'ENGINYERIA

Títol del projecte: LittleProc 2.0. L'Evolució als 16 bits, amb arquitectura superescalar i amb pipeline	
Autor[a]: Roc Prat López	**Data:** Juny de 2011
Tutor[a]/s[es]: Raül Aragonès Ortiz	
Titulació: Enginyeria Tècnica en Informàtica de Sistemes	
Paraules clau (mínim 3) • Català: Multi-nucli, superescalar, pipeline, microprocessador • Castellà: Multi-nucleo, superescalar, tubería, microprocesador • Anglès: Multi-core, superscalar, pipeline, microprocessor	
Resum del projecte (extensió màxima 100 paraules) • Català: Aquest projecte consisteix en evolucionar el LittleProc 1.0, un processador simple dissenyat per a ser destinat en el món de la docència per tres professors de la UAB. Aquestes evolucions consisteixen en aplicar diversos mètodes i arquitectures diferents per tal d'obtenir un millor rendiment del processador, arribant a executar programes amb la meitat de temps que tardava el LittleProc 1.0. Un cop implementades les diferents arquitectures per tal de millorar el rendiment, es realitzarà un estudi de quin tant per cent de millora ha sigut aquest rendiment. • Castellà: Este proyecto consiste en evolucionar el LittleProc 1.0, un procesador simple diseñado para ser destinado en el mundo de la docencia por tres profesores de la UAB. Estas evoluciones consisten en aplicar varios métodos y arquitecturas distintas para obtener un mayor rendimiento del procesador, llegando a ejecutar programas con la mitad de tiempo que tardaba el LittleProc 1.0. Una vez implementadas las diferentes arquitecturas para mejorar el rendimiento, se realizará un estudio de que tanto por ciento de mejora ha sido este rendimiento. • Anglès: This project consists of evolving the LittleProc 1.0, one simple processor designed to be destined in the world of the teaching by three teachers of the UAB. These evolutions consist of applying several methods and different architectures to obtain a major output of the processor, managing to execute programs with the half of time that was late the LittleProc 1.0. Once implemented the different architectures to improve the output there will be realized a study of which so much per cent of improvement has been this output.	

Taula de Continguts

1 Introducció

Amb aquest projecte el que es durà a terme és una millora mitjançant l'ús de diverses arquitectures de computadors del LittleProc 1.0. Un cop implementades totes les arquitectures, s'estudiarà quin és el rendiment d'aquestes i la millora de temps que s'ha comporta cada una respecte el microprocessador inicial. Les dues arquitectures que s'implementaran són la segmentada i la superescalar. Prèviament però, també s'evolucionarà el LittleProc 1.0 dels 8 bits cap als 16 bits, afegint-hi a més a més a aquest nou processador implementat un control per si es produeixen interrupcions i també afegint-hi un registre que ens permeti llegir i escriure dades a l'exterior del processador.

El LittleProc 1.0 és un petit processador de 8 bits que es pot implementar en una placa de prototipatge, aquest va ser dissenyat per el Joaquim Saiz, l'Antoni Portero i el Raül Aragonès, director d'aquest proejcte que es durà a terme. Aquests són un grup de professors de l'Escola d'Enginyeria de la Universitat Autònoma de Barcelona que imparteixen la seva docència en l'àmbit de les enginyeries informàtica i electrònica. El LittleProc va ser dissenyat per a ser utilitzat en l'àmbit docent, sent una molt bona eina de treball pels alumnes i els altres professors que estudien o imparteixen assignatures vinculades al món dels sistemes digitals. Aquest processador utilitza una arquitectura Von Neumann, explicada posteriorment, i una unitat aritmètica lògica. Per tant l'objectiu del projecte, tal i com s'ha dit, és millora el rendiment del LittleProc 1.0 mitjançant diferents arquitectures com pot ser l'arquitectura Harvard o bé l'ús de dos unitats aritmètiques lògiques.

En aquesta memòria es podran veure totes les modificacions aplicades al LittleProc en les diverses arquitectures així com l'estudi realitzat sobre el rendiment de cada una d'elles. Per començar ens trobem amb l'estudi de viabilitat i la planificació que es va realitzar sobre aquest projecte per decidir si era òptim realitzar-lo o no. Seguidament ens trobarem amb una petita introducció a les diferents arquitectures i tècniques que s'utilitzen en

els computadors per tal de millorar el seu rendiment. Posteriorment veurem una petita introducció del LittleProc 1.0, amb una explicació dels components dels que està format i quin es el seu funcionament en termes generals a més a més de llitar les seves principals característiques. Els següents capítols que ens apareixeran serviran per a comprendre el disseny de les diferents arquitectures, des de el pas als 16 bits fins al LittleProc superescalar final. Finalment es trobarà un capítol amb les diferents dades experimentals obtingudes i la realització d'un estudi sobre aquestes.

1.1 Objectius

Els principals objectius d'aquest projecte són:

- Evolucionar el LittleProc 1.0 que és un processador de 8 bits als 16 bits, afegint-li noves instruccions, un control d'interrupcions i un registre que permeti connectar el processador amb l'exterior.
- Realitzar una segmentació en l'anterior evolució, permetent reduir el temps de d'execució d'un programa.
- Finalment utilitzar dos processadors segmentats per tal de poder executar el programa encara amb menys temps.
- Realitzar un estudi de les diverses arquitectures implementades, estudiant els temps d'execució i les millores de rendiment que suposen respecte el processador original.

1.2 Motivacions

El LittleProc va ser estudiat en l'assignatura de Sistemes Digitals II impartida en el segon curs de l'Enginyeria Tècnica en Informàtica de Sistemes. Allà és on vaig descobrir els circuits digitals, una area entre l'informàtica i l'electrònica que hem va captar l'atenció. Estudiant les diferents propostes de PFCs que ofertaven els professors de la UAB vaig veure que n'hi havia una que consistia en evolucionar el LittleProc i hem vaig decidir a realitzar aquest projecte, que penso que també servirà de cara als següents estudis que tinc pensat realitzar.

2 Estudi de viabilitat

2.1 Introducció

En aquest capítol és durà a terme un estudi sobre la viabilitat del projecte a realitzar. El projecte és basa en la modificació d'un microprocessador realitzat l'any anterior en l'assignatura de Sistemes Digitals II, evolucionant-lo mitjançant diferents arquitectures per a realitzar una avaluació de cada una de les modificacions i estudiar els diferents resultats. Els principals objectius del projecte seran descrits seguidament.

2.1.1 Tipologia i paraules clau

El projecte que es realitzarà no es pot incloure a dintre de una sola categoria donat que engloba aspectes de diferents tipologies. Es pot considerar un projecte d'investigació, donat que es durà a terme un bon anàlisi de l'àrea dels processadors per a millorar la comprensió d'aquesta. Un cop desenvolupat el projecte es valorarà les diferents alternatives per a identificar els punt forts i els punts febles de cada una.

També es pot considerar un projecte d'avaluació, donat que com s'ha explicat anteriorment, un cop trobades les diferents implementacions del processador, s'avaluarà quina és la millor implementació, amb els avantatges i desavantatges que comporta cada una.

En el projecte podem trobar les següents paraules clau, que es definiran més endavant:

- **Superscalar**
- **Pipeline**
- **Processador**
- **Multi-core**

2.1.2 Descripció

La idea de realitzar aquest projecte va sorgir d'una proposta del professor de Sistemes Digitals II de potenciar el processador LittleProc 1.0 per a realitzar un seguit de millores que el duguessin a aproximar-se als processadors que hi ha actualment al mercat i obtenir un processador d'alt rendiment embegut dins de una FPGA, fet que actualment no es troba en el mercat.

El LittleProc 1.0 és un processador senzill que executa les instruccions seqüencialment, una després de l'altre. El que es durà a terme durant el projecte serà anar millorant-lo pas per pas, primerament introduint-li un pipeline que ens permeti executar més d'una instrucció per cicle. Posteriorment es crearà un processador superscalar que ens permetrà implementar un multi-core, on funcionarà més de un processador al mateix temps.

Aquest projecte es desenvoluparà sobre un context econòmic favorable, donat que tot el material utilitzat es pot obtenir gratuït, o bé ja es troba al departament del cap del projecte.

2.1.3 Objectius del projecte

Els objectius principals del projectes ja han estat anomenats anteriorment, però seguidament seran descrits amb més detall:

El primer objectiu ha realitzar serà millorar el anterior processador augmentant el nombre de bits en que podrà operar aquest. També s'augmentarà el número de instruccions del LittleProc 1.0, afegint-hi operacions lògiques i operacions aritmètiques com la divisió i el residu entre d'altres. Totes aquestes ampliacions comportarà canvis a tots els conjunts que formen el processador com la UC i la UP. Aquest objectiu hauria d'estar assolit a mitjans de Octubre.

El segon objectiu que s'ha de complir és afegir un pipeline al processador modificat anteriorment a fi de poder realitzar més d'una operació en un cicle. Aquest fet ens comportarà una millora substancial en el temps d'execució de qualsevol programa que vulguem executar en el nostre processador. Aquesta part del projecte s'ha de realitzar fins a mitjans de Gener.

El tercer objectiu que s'ha de dur a terme és el de transformar el processador amb pipeline a un superscalar. Aquest fet ens permetrà utilitzar la tecnologia multicore, amb més de un processador funcionant a la vegada, fet que ens reduirà substancialment el temps d'execució de qualsevol programa. A finals d'Abril aquesta part hauria d'estar enllestida.

Prioritització dels objectius		
ID	**Objectiu**	**Tipus**
1	Ampliació nombre de bits i nombre d'instruccions del LittleProc 1.0	Crític
2	Creació d'un pipeline en el processador	Crític
3	Introducció de l'arquitectura superscalar i desenvolupament d'un processador multicore	Secundari
4	Avaluació dels diferents tipus de processadors desenvolupats	Crític

Taula 2.1 Prioritització dels objectius

2.1.4 Definicions, acrònims i abreviacions

Seguidament es procedirà a realitzar un sèrie de definicions bàsiques per a poder entendre termes del nostre projecte:

. *Superscalar:* És el terme utilitzat per a designar un tipus de microarquitectura de processador capaç d'executar més de una instrucció per cicle de rellotge. Normalment es té el grau de superescalaritat, que ens indica quantes instruccions poden entrar al processador alhora per cicle.

.*Pipeline:*Aquesta arquitectura consisteix en anar transformant un flux de dades en un procés comprès per varies fases seqüencials, sent l'entrada de cada una la sortida de l'anterior.

.UP: És la unitat de procés d'un processador. Aquesta té la funció d'emmagatzemar el conjunt de registres que s'encarreguen del correcte funcionament de la execució d'instruccions, així com de realitzar les operacions necessàries.

.UC: És la unitat de control del processador. La unitat de control és l'encarregada de generar la seqüència d'ordres necessàries per a executar cada instrucció emmagatzemada a la RAM.

.RAM: És la memòria d'accés aleatori. És la memòria des de on el processador rep les instruccions i guarda els resultats

.ROM: És la memòria només de lectura. En ella s'emmagatzemen totes les instruccions del processador que s'utilitzaran per a executar els programes.

.Multi-core: És un processador que combina dos o més processadors independents en un sol paquet.

.Quartus II: Serà el software utilitzar per a poder realitzat i simular el projecte abans d'implementar-lo en una PLA. Aquest software és gratuït i és de la companyia Altera

PLA: Una PLA és un dispositiu lògic programable. PLA significa Array Lògic Programable. Aquesta és un circuit que es pot programar per a executar una funció complexa. Normalment s'utilitzen per a implementar lògica combinacional, però alguns PLA es poden usar per a implementar dissenys lògics seqüencials.

FPGA: Una FPGA és un dispositiu semiconductor que conté blocs de lògica que permeten ser programats. La lògica programable pot reproduir des de funcions tan senzilles com les que realitza una porta lògica fins a sistemes complexos en un xip.

2.1.5 Parts interessades

En el projecte a realitzar està implicat el Raúl Aragonés, del departament de microelectrònica i sistemes electrònics de la UAB, fent alhora la funció de cap del projecte i de director del projecte. Apart d'aquests dos càrrecs també se'l pot considerar coordinador o líder del projecte donat que ell marca la pauta de com s'han de fer les coses . Ell és professor de la UAB des de ja fa uns anys i va ser un dels implicats en el disseny del LittleProc 1.0, que és el processador a potenciar. La seva principal responsabilitat serà la de controlar el desenvolupament del processador i supervisar el correcte funcionament de cada una d'elles.

Per altre part el implicat soc jo, que m'encarregaré de desenvolupar totes les altres variants del processador i provar-les perquè funcionin correctament.

2.1.6 Referències

En quan a normatives a seguir, la que regeix aquest projecte és la normativa de la UAB alhora de realitzar els projectes finals de carrera. Segons aquesta normativa, els professors de la UAB que han dirigit o coordinat treballs són coautors de l'obra final, juntament amb l'estudiant, en l'anomenada propietat intel·lectual. A més a més, i segons la normativa esmentada, totalitat dels drets d'explotació de l'obra corresponen a la universitat i en cas de suposar un benefici econòmic, l'autor o conjunt d'autors tindran dret a una participació del 50% dels beneficis nets obtinguts, segons els drets d'explotació.

Una altre normativa que regirà el projecte serà la de privacitat de dades. Finalment la normativa que regeix el software que s'utilitzarà per realitzar el projecte fi de carrera serà la copyright.

2.1.7 Producte i documentació del projecte.

Durant el desenvolupament del projecte, s'obtindran diferents tipus de processadors, amb diferents arquitectures cada un.
Un cop realitzat el projecte es realitzarà un estudi de les diferents arquitectures de processadors, des de un processador simple a un multi-core, amb els diferents temps d'execució analitzats i justificats. Es podrà veure l'evolució que han tingut els processadors als llarg de la història, amb les corresponents millores aportades en cada etapa, fet que servirà com a ajuda per comprendre el funcionament de les diferents arquitectures de processadors.

Per tant l'objectiu final del projecte serà presentar una evolució arquitectural i en rendiment d'un processador encastable.

2.2 Estudi de la situació actual

2.2.1 Context

Actualment, el processador a partir del que hem de desenvolupar les nostres millores és un processador simple. Aquest està format per una UP, formada per registres, una ALU i 2 busos per on passen les dades. Una UC, on hi trobem la ROM amb totes les microinstruccions codificades i una RAM.

En la UP hi trobem 3 registres que s'utilitzen per emmagatzemar els resultats de les operacions i per a poder operar amb elles. També podem trobar el registre IR, on s'emmagatzema la direcció cap a on es produirà un salt. El registre PC, encarregat d'emmagatzemar l'adreça física de la següent instrucció a ésser executada. El registre MAR, encarregat d'emmagatzemar l'adreça de la posició de memòria (RAM) on es troba una dada que hem de recollir . El registre MDR, que s'encarrega d'emmagatzemar el contingut d'una adreça d'una posició de memòria interna o bé una dada que ve pel bus 2 (**ALU**). I finalment en quant a

registres també trobem el ACC, encarregat d'emmagatzemar temporalment la dada d'un operand de l'ALU.

En la UP també hi ha una de les parts més importants del processador, la ALU, que és l'encarregada de realitzar totes les operacions necessàries per al correcte funcionament del processador. Aquestes operacions inicialment són les de pas transparent, suma, resta, màscara dels 4 primers bits, incrementador d'una unitat i desplaçament a la dreta/ esquerra de bits.

En general, i tal i com hem dit anteriorment la UP és la unitat del processador que té la funció d'emmagatzemar el conjunt de registres que s'encarreguen del correcte funcionament de la execució d'instruccions, així com de realitzar les operacions necessàries.

A part de la UP en processador també podem trobar la UC. Aquesta és l'encarregada de generar la seqüència d'ordres necessàries per a executar cada instrucció emmagatzemada a la RAM, i està formada per un seqüenciador, que té per objectiu entregar a la seva sortida l'adreça de la següent microinstrucció a executar. Una ROM per a emmagatzemar les microinstruccions, formada per 25 bits. A més a més d'un multiplexor, un descodificador de les instruccions, una xarxa de busos i una lògica combinacional senzilla.
Finalment, en el processador que tenim realitzat actualment des de el curs anterior podem trobar la RAM, que és la memòria a on està guardat el programa que després s'executarà en el processador.

Des de una visió més global del context actual en general, existeixen ja molt processadors superscalars, essent actualment el darrer avanç, el processadors multi-core de 4 nuclis. Però s'ha de tenir en compte que la finalitat del nostre projecte a part d'analitzar les diferents arquitectures serà la d'obtenir un processador d'alt rendiment que estigui embegut dins d'una FPGA, aquest fet no és produeix actualment. Si ho mirem des de el punt de vista de la integració, aquest fet és molt important donat que actualment en el mercat no se'n troba i el que fan els dissenyadors habitualment quan

volen implementar al seu disseny digital és comprar-se un processador ja fabricat i afegir-li al costat la FPGA amb el disseny digital. Aquest pas s'anomena hard-soft.

2.2.2 Lògica del sistema

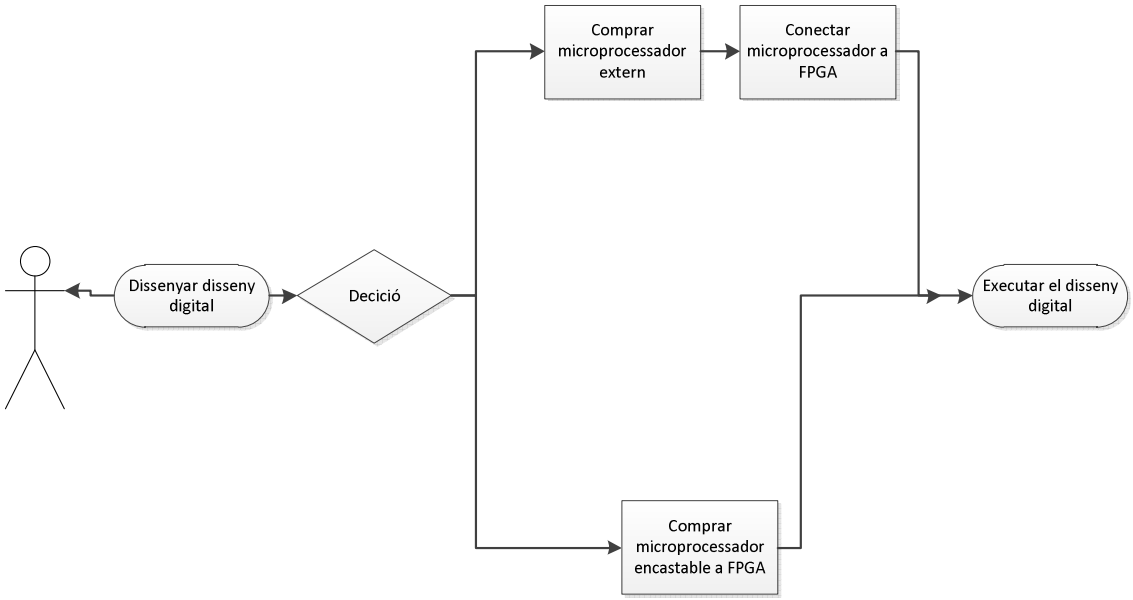

Figura 2.1 Lògica del sistema

En el sistema actual, el dissenyador només pot comprar microprocessador no encastables per tant només pot seguir el camí superior. L'objectiu d'aquest projecte és permetre al dissenyador poder comprar un processador que el pugui encastar a la seva FPGA i augmentar el grau d'integració.

2.2.3 Descripció física

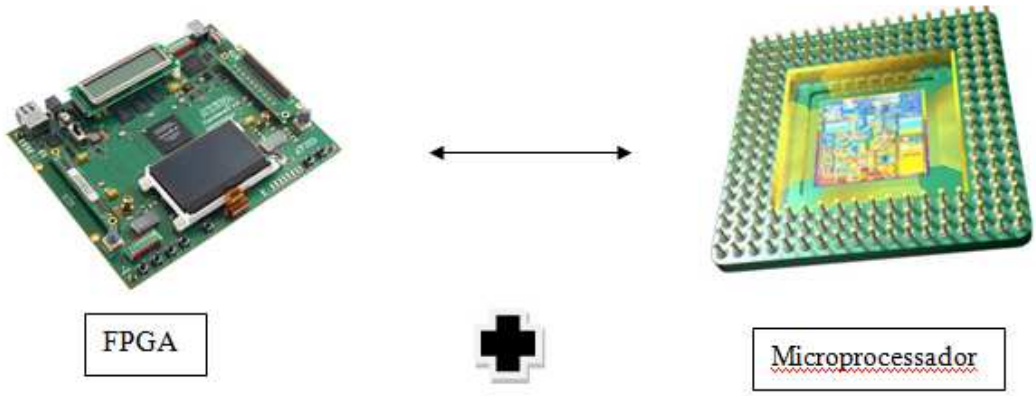

Figura 2.2 Descripció física

Actualment en el mercat no trobem microprocessadors que es puguin encastar a dintre de una FPGA. Per tant el que fan els dissenyadors digitals és comprar-se el microprocessador a part per a connectar-lo a la FPGA.

2.2.4 Usuaris i/o personal del sistema

En el sistema intervindrà el dissenyador digital que s'encarregarà d'utilitzar el microprocessador que nosaltres dissenyem, encastant-lo a la seva FPGA per a així poder utilitzar el seu disseny. Així s'evitarà comprar un microprocessador extern, podent plantejar-se el seu disseny amb un elevat grau d'integració.

2.2.5 Diagnòstic del sistema

Deficiències:

- En el sistema que hi ha actualment, el dissenyador a de comprar el microprocessador apart de la seva FPGA.
- Cost elevat del microprocessador
- Poc grau de integració
- Microprocessadors amb un elevat temps d'execució

Millores:

- El dissenyador podrà encastar el microprocessador a la FPGA.
- S'aconseguirà un molt bon grau d'integració.

- Es millorarà substancialment els temps d'execució dels microprocessadors

2.3 Requisits del sistema

2.3.1 Requisits funcionals

Seguidament es formularà una llista amb els requisits funcionals del nostre sistema ordenats per importància:

El primer requisit funcional que trobem és el d'ampliar el número de instruccions del LittleProc 1.0, donat que aquest és un processador senzill que incorpora poques operacions. A més a més el que interessat en el mercat és tenir un bon processador amb el número màxim d'operacions ja dissenyades. Tal i com s'ha comentat anteriorment, en aquesta fase s'incorporaran operacions com la multiplicació, la divisió, el residu, operacions lògiques com AND, OR ... I diferents tipus d'adreçament de dades. Aquest també és un requisit essencial donat que a partir d'aquest també s'han de desenvolupar les altres arquitectures que tindrà el nostre processador.

L'objectiu proposat d'obtenir un processador pipeline també el podem considerar com a un requisit funcional de caire essencial, donat que serà la primera evolució seria obtinguda del LittleProc 1.0. A més a més la següent arquitectura a realitzar (superscalar) està subjecte a aquest requisit.

Dintre dels requisits condicionals trobem el de realitzar el nostre processador amb arquitectura superscalar, amb més d'un processador funcionant a la vegada, ja aquest objectiu estarà subjecte a que haguem pogut acomplir l'objectiu d'obtenir un processador amb pipeline, i que aquest obtingut, es pugui millorar més.

2.3.2 Requisits no funcionals

El primer requisit no funcional a complir, serà el de tenir el processador augmentant de 12 a 16 el número de bits amb que aquest podrà operar respecte de l'anterior. Aquest requisit és essencial donat que a partir d'aquesta evolució del processador és realitzaran las posteriors. Es considerà requisit no funcional perquè en ell es troba una limitació de rendiment del processador a dissenyar, restringint a 16 el número de bits en que podem operar.

En el projecte ha realitzar no es troben molts requisits no funcionals donat que la base d'aquest és desenvolupar el LittleProc 1.0 per a poder obtenir diverses millores del temps d'execució sense tindre especificat ningun resultat a assolir com poguessin ser temps d'execució...

2.3.3 Restriccions del sistema

En aquest punt del estudi de viabilitat s'analitzarà totes les possibles restriccions que ens trobem en el nostre sistema.

Una restricció és la definida per el meu director del projecte, i definida anteriorment com a requisit no funcional. Aquesta és la restricció imposada de que el número de bits sigui 16, no podent ser un número més elevat.

Un altre de les restricció existents, i evidentment obvia, és la restricció temporal que existeix, donat que el projecte s'ha de tindre realitzat abans de un determinat dia, el dia en que s'entregarà la memòria d'aquest.

2.3.4 Catalogació i priorització dels requisits

Requisit	Funcional	Objectiu	Priorització
Ampliar número d'instruccions del LittleProc 1.0	Sí	1	Essencial
Obtenir el processador amb l'arquitectura	Sí	2	Essencial

pipeline			
Anàlisis i estudi dels diferents temps d'execució obtinguts amb cada una de les arquitectures trobades	Sí	5	Essencial
Ampliació del nombre de bits del LittleProc 1.0 a 16 bits	No	1	Essencial
Obtindre un processador amb arquitectura superscalar	Sí	3	Condicional

Taula 2.2 Prioritització dels requisits

2.4 Alternatives i selecció de la solució

Com s'ha anomenat anteriorment, en el mercat no es troba cap processador pipeline, superscalar o multi-core que es pugui incorporar dins de una FPGA.

2.4.1 Alternativa 1

La primera alternativa que trobem i que és la que utilitzen la major part dels dissenyadors de disseny digital, és comprar un microprocessador ja fabricat i li afegeix al costat la FPGA amb el disseny hardware. Aquesta alternativa es pot veure que no és econòmica donat que el preu d'un microprocessador ja fabricat en el mercat és bastant elevat. A més s'ha de tenir en compte que els microprocessadors fabricats són simples, i el que es vol normalment siguin ràpids en donar la resposta dels programes executats, per tant, que tinguin el menor temps d'execució possible.

2.4.2 Alternativa 2

La segona alternativa que trobem, i que és la que proposem el meu cap de projecte i jo, és la de dissenyar el processador per a poder encastar-lo a la FPGA directament. Aquest fet, i des de el punt de vista de la integració, és molt important. S'ha de tindre en compte que es dissenyarà un processador

amb diferents arquitectures, per tant es podrà reduir el temps d'execució d'un processador simple que ja es troba en el mercat. A més a més, es té per objectiu presentar una evolució arquitectural i en rendiment del processador encastable.

2.4.3 Solució proposada

La solució proposada és l'alternativa 2, donat que així es podran obtenir processadors més ràpids i amb la possibilitat d'encastar-los a la FPGA, fet molt important des de el punt de vista de la integració. A més a més, i tal i com s'ha comentat anteriorment, econòmicament parlant, comprar un microprocessador extern a la FPGA és bastant més car que fer-ne un que després es pugui encastar.

2.5 Conclusions

Un cop realitzat el estudi de viabilitat es poden extreure diverses conclusions positives d'aquest. Tal i com hem vist, actualment en el mercat només podem trobar microprocessadors simples i no encastables a una FPGA. Aquest fet porta als dissenyadors digitals a tenir que comprar el microprocessador i tindre'l que connectar a la FPGA a on volen implementar el seu disseny.

A part de realitzar un microprocessador encastable, el que també es millorarà serà el temps d'execució d'aquest, donat que al implementar-lo amb diferents arquitectures, s'anirà reduint el temps. Per tant els dissenyadors que necessitin el processador, el podran encastar a la seva FPGA i a més a més obtindran una millora substancial del temps d'execució.

Finalment, i després del corresponent anàlisi de cada arquitectura, es podran extreure conclusions de quina és la millor arquitectura segons els temps d'execució obtinguts.

Econòmicament parlant, també veiem que el projecte és viable, donat que no hi haurà despeses. El software que s'utilitzarà per a dissenyar els

diversos tipus de microprocessador el podem obtenir gratuïtament des de la web oficial.

3 Planificació del projecte

3.1 Introducció

En aquest document és durà a terme la realització de la planificació del projecte fi de carrera, comentant cada fase que aquest tindrà. També es planificaran totes les activitats i recursos que intervindran en el projecte.

3.1.1 Descripció

Aquest document recull el Pla del projecte, és a dir, el conjunt d'activitats que permeten desenvolupar, executar i controlar el projecte. El document inclou les tasques i punts de control del projecte, els recursos del projecte, el calendari del projecte, l'avaluació de riscos i el pressupost del projecte. S'ha utilitzat un metodologia lineal en l'elaboració del pla.

Per a realitzar la planificació del projecte, s'emprarà la eina de Microsoft Project, que s'ha estudiat a l'assignatura de MiGP.

3.1.2 Definicions, acrònims i abreviacions

Seguidament s'explicaran un seguit de definicions de termes, acrònims i abreviacions que apareixeran en aquest document perquè quedin clares.

- *Microsoft Project*: programa de Microsoft utilitzat per la gestió de projectes.
- *TRAC*: Eina per la gestió i control de projectes.
- *WBS*: Work Breakdown Structure.
- *Milestone*: Punt de control.
- *Diagrama de Gantt*: Cronograma del projecte.

3.1.3 Referències

La normativa que es farà servir per a realitzar aquest document és la de la UAB. A part també hi intervindran els següents reglaments i normatives que es poden trobar a les següents webs:

1. Microsoft project → http://www.microsoft.com/project/
2. TRAC → http://trac.edgewall.org/

3.2 WBS (Work Breakdown Structure)

En aquest apartat del document es llistaran i descriuran totes les fases, activitats i tasques del projecte.

3.2.1 Fases i activitats del projecte

Fases	Descripció
Iniciació	Fase d'iniciació. Inclou les activitats definició del projecte, assignació i matriculació.
Planificació	Inclou l'Estudi de Viabilitat i el Pla del Projecte. En aquesta fase, també és durà a terme l'anàlisi dels requisits funcionals i no funcionals. Finalment també es farà el disseny del projecte
Desenvolupament	Fase de desenvolupament de l'aplicació.
Test i proves	Fase de prova del sistema. Inclou tests unitaris Test i proves i d'integració. En aquesta part es duran a terme els tests de cada arquitectura mitjançant el software.
Implantació	L'aplicació s'instal·la en el seu entorn real, la FPGA. Un cop implementat en la FPGA, es duran a terme tot un seguit d'anàlisi dels temps d'execució de cada arquitectura.
Generació de documents	Fase de documentació del projecte. Inclou manuals i memòria del projecte.
Tancament del projecte	Fase de tancament. El director del projecte signa l'acceptació i tancament del projecte.
Defensa del projecte	Defensa del projecte davant la comissió. Es durà a terme l'exposició oral del projecte

Taula 3.1 Fases del projecte

3.2.2 Diagrama WBS

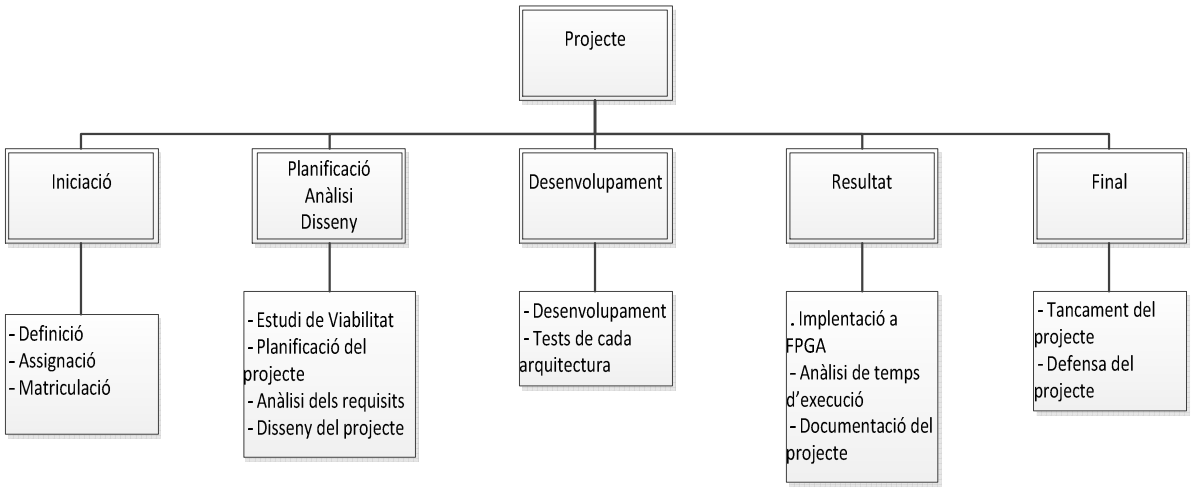

Figura 3.1 Diagrama WBS

3.2.3 Milestones

Nom	Descripció	Data
Iniciació	Matriculació	29/10/2010
Est. Viabilitat	Aprovació	11/11/2010
Pla del Projecte	Aprovació	04/11/2010
Anàlisi	Aprovació	25/11/2011
Disseny	Aprovació	15/02/2011
Tancament	Acceptació	20/06/2011
Defensa	Avaluació	06/07/2011

Taula 3.2 Milestones

3.3 Recursos del projecte

En aquest apartat es valoraran els recursos humans i materials assignats a cada tasca del projecte.

3.3.1 Recursos del projecte

La valoració que es posarà sobre cada recurs és una aproximació donat que realment ningun recurs cobrarà cap quantitat.

Recurs humà	Valoració
Director de projecte	90 €/h
Cap de projecte	50 €/h
Analista	45 €/h
Tècnic de proves	20 €/h
Programador/Desenvolipador	35 €/h

Taula 3.3 Costs dels recursos humans del projecte

S'ha de tenir en compte que la persona que fa la funció de director de projecte en el meu projecte final de carrera és el meu professor, en Raúl Aragonés.
Per altra banda, jo seré la persona que farà de analista, tècnic de proves, cap de projecte i programador/desenvolupador a la vegada.
Respecte als recursos materials, s'utilitzaran els recursos disponibles en el departament com la FPGA. A més a més tot el desenvolupament del projecte es farà utilitzant programari de domini públic.

3.3.2 Calendari dels recursos

Seguidament es descriuran els recursos humans que s'utilitzaran durant tot el projecte amb les fases i activitats que fa cada un d'ells:

Cap de projecte: Iniciació, Planificació, Generació de documents, tancament i defensa. Punts de control.
Analista: Anàlisi i disseny, Implantació i Punts de control d'anàlisi, disseny i desenvolupament.
Programador: Disseny, Desenvolupament i Test. Parcialment en l' implantació.
Tècnic de proves: Fase de test.

Els recursos materials s'utilitzaran principalment en les fases de desenvolupament, test i implantació. El software per a desenvolupar l'aplicació per exemple i posteriorment en l'etapa d'implantació s'utilitzarà la FPGA.

3.4 Calendari del projecte

El projecte final de carrera començarà a l'Octubre de 2010 i acabarà el Juliol de 2011. En ell s'hi dedicarà un total de 11 hores setmanals, treballant 3 hores els dilluns, dimarts i dijous, i solament una hora els dimecres i els divendres. La durada del projecte serà de 44 dies treballats. S'ha de considerar el dies festius corresponents al 6 i 8 de Desembre a més a més de les vacances de Nadal del 24 al 7 de gener i de les de setmana santa.

Data de començament: 29 d'Octubre
Data de finalització: 6 de Juliol
Eines de planificació i de control: Microsoft Project i Trac

3.4.1 Dependències

En el nostre projecte, podem considerar que utilitzarem un model lineal, donat que no es desenvoluparà una part del projecte fins que no s'hagi acabat l'anterior.

Tot i que podem considerar que totes les fases seran independents menys la primera. La primera consisteix en evolucionar el microprocessador de 12 a 16 bits, i a més a més augmentar el número de instruccions del processador. Aleshores a partir d'aquesta fase es desenvoluparan totes les altres, utilitzat com a base la que s'acaba d'anomenar.

En la fase de desenvolupament es preveu un model àgil de tal manera que el disseny, el desenvolupament i el tes segueixin un model iteratiu.

La fase de generació de documents es realitzarà al final perquè inclourà documents elaborats durant el desenvolupament del projecte, tals com el estudi de viabilitat, el pla del projecte...

3.4.2 Quadre de tasques del projecte

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
Projecte fi de carrera	**44 días**	**vie 29/10/10**	**mié 06/07/11**		
Inici del projecte: assignació i matriculació del projecte	1 hora	vie 29/10/10	vie 29/10/10		Cap de projecte;Director de projecte
Planificació	**2,5 días**	**vie 29/10/10**	**jue 11/11/10**		
Anàlisi de l'aplicació	**3,88 días**	**vie 05/11/10**	**jue 25/11/10**		
Anàlisi de requisits (funcionals i no funcionals)	10 horas	vie 05/11/10	jue 11/11/10	8	Analista
Anàlisi de dades	5 horas	jue 11/11/10	lun 15/11/10	10	Analista
Anàlisi de seguretat i legalitat	5 horas	mar 16/11/10	jue 18/11/10	11	Analista
Documentació de l'anàlisi	10 horas	jue 18/11/10	mié 24/11/10	12	Analista
Aprovació de l'anàlisi	1 hora	jue 25/11/10	jue 25/11/10	13	Cap de projecte;Analista;Director de projecte
Disseny de l'aplicació	**12,88 días**	**jue 25/11/10**	**mar 15/02/11**		
Augment de el número de bits del LittleProc 1.0	20 horas	jue 25/11/10	jue 09/12/10	14	Analista;Desenvolupador
Augment de les instruccions del LittleProc 1.0	20 horas	vie 10/12/10	jue 23/12/10	16	Analista;Desenvolupador
Disseny arquitectura pipeline	20 horas	jue 23/12/10	mié 19/01/11	17	Analista;Desenvolupador
Disseny arquitectura superscalar i multicore	40 horas	jue 20/01/11	mar 14/02/11	18	Analista;Desenvolupador
Documentació dels diferents dissenys	2 horas	mar 15/02/11	mar 15/02/11	19	Analista
Aprovació del disseny	1 hora	mar 15/02/11	mar 15/02/11	20	Cap de projecte; Analista ;Director de projecte
Desenvolupament de l'aplicació	**16,25 días**	**mié 16/02/11**	**mié 18/05/11**		
Preparació de l'entorn de desenvolupament	2 horas	mié 16/02/11	jue 17/02/11	21	Desenvolupador
Desenvolupament dels augments del LittleProc 1.0	36 horas	jue 17/02/11	vie 11/03/11	23	Desenvolupador
Desenvolupament de l'arquitectura pipeline	36 horas	lun 14/03/11	lun 04/04/11	24	Desenvolupador
Desenvolupament de l'arquitectura superscalar i multicore	56 horas	mar 05/04/11	jue 18/05/11	25	Desenvolupador
Test i proves	**16,75 días**	**jue 17/02/11**	**lun 23/05/11**		
Prova del augments del LittleProc 1.0 en el software	2 horas	jue 17/02/11	jue 17/02/11	22	Tècnic de proves; Desenvolupador
Prova de l'arquitectura pipeline en el software	2 horas	mar 05/04/11	mar 05/04/11	23	Tècnic de proves; Desenvolupador

Prova de l'arquitectura superscalar i multi-core en el software	4 horas	jue 18/05/11	jue 18/05/11	24	Tècnic de proves ;Desenvolupador
Estudi de les diferents proves realitzades	2 horas	vie 20/05/11	vie 20/05/11	30	Tècnic de proves; Desenvolupador
Aprovació de les diferents proves realitzades	2 horas	lun 23/05/11	lun 23/05/11	31	Tècnic de proves; Desenvolupador; Cap de projecte; Director de projecte
Generació de documents (memòria del projecte)	36 horas	vie 27/05/11	lun 20/06/11	32	Cap de projecte;Analista
Tancament del projecte	1 hora	lun 20/06/11	lun 20/06/11	33	Cap de projecte; Director de projecte
Defensa del projecte	1 hora	mié 06/07/11	mié 06/07/11		Cap de projecte

Taula 3.4 Quadre de tasques del projecte

3.4.3 Calendari temporal

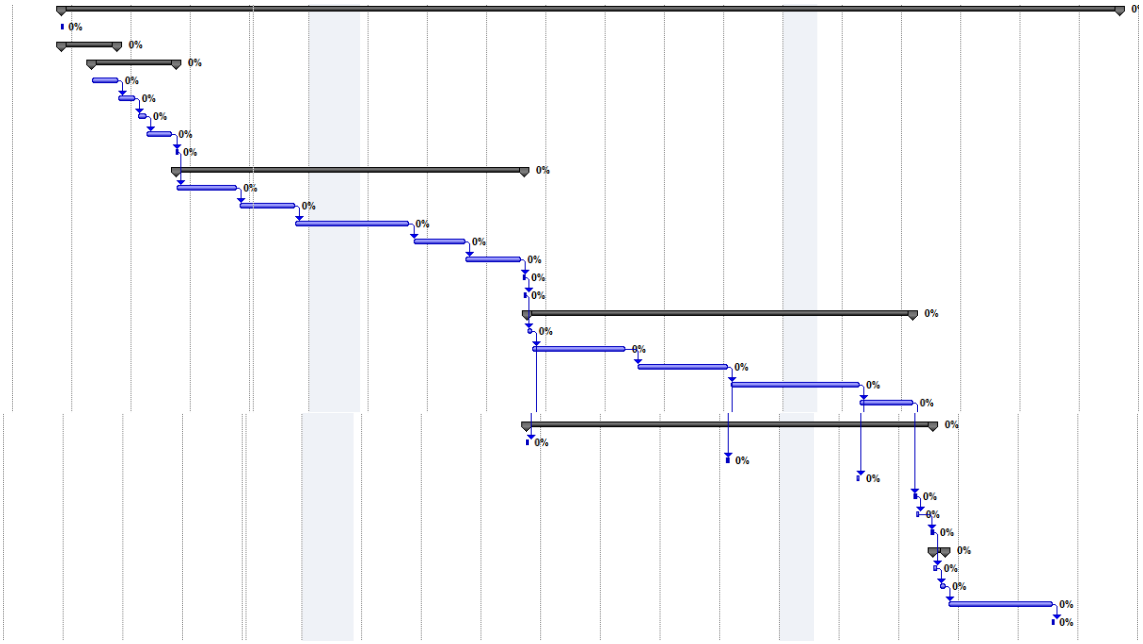

Figura 3.2 Calendari temporal del projecte

3.5 Avaluació de riscos

3.5.1 Llista de riscos

R1. Planificació temporal optimista: Pla de projecte. No s'acaba en la data prevista, augmenten els recursos.

R2. Manca alguna tasca necessària: Pla de projecte. No es compleixen els objectius del projecte. Aquest fet passa quan s'ha deixat de planificar alguna tasca important.

R3. Canvi de requisits: estudi de viabilitat, anàlisi. Endarreriment en els desenvolupament i resultat.

R4. Dificultat per accedir als stakeholders: estudi de viabilitat, anàlisi, proves, formació. Manquen requisits o són inadequats, endarreriments, insatisfacció usuaris.

R5. No es fa correctament la fase de test: desenvolupament, implantació. Manca de qualitat, deficiències en l'operativa, insatisfacció usuaris, pèrdua econòmica.

R6. Incompliment d'alguna norma, reglament o legislació: en qualsevol fase. No es compleixen els objectius, repercussions legals.

R7. Manca d'adopció de mesures de seguretat: estudi de viabilitat, anàlisi, desenvolupament. Pèrdua d'informació, incompliment legal, pèrdues econòmiques.

R8. Abandonament del projecte abans de la finalització: en qualsevol fase. Pèrdues econòmiques, frustració.

3.5.2 Catalogació de riscos

	Probabilitat	Impacte
R1	Baixa	Crític
R2	Baixa	Crític
R3	Mitjana	Catastròfic
R4	Baixa	Crític
R5	Mitjana	Crític
R6	Baixa	Marginal
R7	Baixa	Marginal
R8	Mitjana	Catastròfic

Taula 3.5 Catalogació de riscos

3.5.3 Pla de contingència

	Solució
R1	Ajornar alguna funcionalitat, afrontar possibles pèrdues, fer una assegurança. Tornar a realitzar any següent
R2	Revisar el Pla de Projecte, modificar la planificació.
R3	Ajornar funcionalitat, modificar planificació i pressupost.
R4	Fixar un calendari de reunions amb el professor, millorar el contacte amb ell.
R5	Assegurar-se millor en realitzar la fase de test.
R6	Revisar les normes i legislació, consultar un expert, afrontar possibles repercussions penals.
R7	Revisar la seguretat en cada fase, aplicar polítiques de seguretat actives.
R8	No es pot solucionar.

Taula 3.6 Pla de contingència

3.6 Pressupost

En aquest apart es durà a terme un estudi del pressupost que costarà tot el projecte en global, estudiat en diferents subapartats.

3.6.1 Estimació cost de personal

En aquest cas el personal que treballa en aquest projecte no cobrarà cap sou, per tant el cost del personal assignat al projecte és de 0 €/h.

3.6.2 Estimació cost dels recursos

	Cost amortització	Cost unitari	Període amortització	Període utilització
PC del dissenyador, programador, desenvolupador	166,66 €	1000 €	48 m	8 m
Amortització MSOffice	116,5 €	699 €	48 m	8 m
Amortització MSProject	216,66 €	1300 €	48 m	8 m

Taula 3.7 Estimació del cost dels recursos materials

S'ha de tenir en compte que el software utilitzat per a desenvolupar els processadors és pot obtenir gratuïtament des de la web de la companyia en propietat del programa, per tant el cost que tindrà és 0 €.

Finalment explicar que el preu de la llicencia de MSProject realment no l'hem de considerar en aquest pressupost donat que s'obté gratuïtament per un any, (més del temps que durarà el desenvolupament del projecte) des de la pàgina web msdn si ets estudiant de la UAB.

Per tant el valor total que ens costarà realitzar el projecte serà de 499,82 €.

3.6.3 Resum i anàlisi cost benefici

Cost	Valor
Desenvolupament projecte	0 €
Amortització material	499,82 €
TOTAL	499,82 €

Taula 3.8 Cost total del projecte

Podem veure que el projecte costa 499,82 €. Però s'ha de tenir en compte que el ordinador a utilitzar per a realitzar-lo és el personal, no ha estat comprat per a realitzar el projecte, a més a més s'ha pogut obtenir versions gratuïtes del MSOffice. Per tant el cost real del projecte serà de 0 €.

3.7 Conclusions

Segons el que s'ha pogut anar veient en aquest document de la planificació del projecte es pot considerar de que el projecte és viable.

Tal i com s'ha vist el calendari de cada activitat està molt ben definit i per tant no ha d'haver-hi problema al realitzar cada tasca segons quan li toqui. A més a més es pot veure que cap recurs del projecte no està sobre assignat en cap tasca i només a de realitzar 3 hores al dia 3 dies a la setmana i posteriorment 1 hora el dimecres i el divendres, tenint festa el cap de setmana i les vacances corresponents.

Segons aquesta planificació també podem considerar el projecte viable econòmicament segons la relació de cost- benefici, donat que com hem vist el projecte en si no tindrà cap cost. Tampoc no s'espera que tingui cap benefici, però si futurament apareix algun la relació sempre serà positiva. Per tant aquest pla de projecte es pot considerar aprovat.

4 Introducció a les tècniques de funcionament en paral·lel del processadors

Tal i com s'ha comentat anteriorment, existeixen diverses tècniques que ens permeten segmentar i alhora fer funcionar varis processadors al mateix temps. Com que l'objectiu principal del projecte és poder millorar el rendiment d'un processador ja dissenyat, es procedirà a definir varies de les tècniques que existeixen per a poder realitzar una òptima segmentació.

4.1 Arquitectures de memòria

Primerament dir que la memòria també es pot segmentar. Hi ha dos tipus d'arquitectures que utilitzen de diverses formes la memòria:

4.1.1 Arquitectura Von Neumann

La primera arquitectura que s'explicarà és l'anomenada arquitectura Von Neumann, sorgida en l'any 1945 i que rebé el nom del matemàtic búlgar John Von Neumann que la va dissenyar per primera vegada.

Aquesta arquitectura es bassa en connectar el processador a una sola memòria. On aquest dispositiu d'emmagatzematge fa alhora de memòria de programa (memòria on hi ha les instruccions del programa a executar) i memòria de dades (memòria on hi ha les dades que s'utilitzaran per a executar el programa i on es guardaran les dades noves). N esquema d'aquesta arquitectura seria el següent:

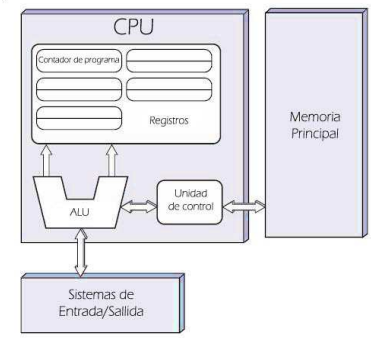

Figura 4.1 Arquitectura Von Nemann

On podem veure que només hi ha un sola memòria connectada a la CPU. Un dels principals inconvenients en utilitzar aquesta arquitectura és que es produeix un alentiment degut a que les instruccions de la memòria junt amb les dades han de passar per el mateix bus, produint-se l'anomentat coll de botella de Von Neumann.

4.1.2 Arquitectura Harvard

Per altra banda tenim l'arquitectura Harvard. Aquesta va ser dissenyada per primera vegada en la computadora Harvard Mark I. El seu funcionament es bassa en diferencia de la anterior en tenir dues memòries separades, una per a la memòria de dades i l'altre per a la memòria d'instruccions. El seu esquema seria el següent:

Figura 4.2 Arquitectura Harvard

Podem veure clarament que connectades a la Unitat de Control hi ha dos memòries diferents, la de dades i la de instruccions. Però aquesta arquitectura té l'inconvenient de que només és millora el rendiment quan el flux de instruccions i de dades és més o menys el mateix.

Com més endavant es veurà, en el disseny de les diferents arquitectures del processador s'utilitzaran les dues arquitectures de memòria vistes.

4.2 Mètodes de millora del rendiment d'un processador

Per altre banda existeixen diversos mètodes per a augmentar el rendiment d'un processador, varis d'aquests es poden veure a continuació:

4.2.1 Hyperthreading

Un dels mètodes més destacat és el Hyperthreading. Aquesta marca va ser registrada per la empresa Intel per a nombrar la seva implementació de la tecnologia Multithreading Simultani, també coneguda com a SMT. Aquesta permet als programes preparats per a executar múltiples fils (multi-threaded) processar-los en paral·lel dins d'un únic processador, incrementant l'ús de les unitats d'execució del processador. Aquesta tecnologia consisteix en simular dos processadors lògics dins d'un únic processador físic, millorant el rendiment del processador, donat que al simular dos processadors es poden aprofitar millor les unitats de càlcul mantenint-les ocupades durant més temps. Segons Intel això provoca una millora de entre 15% i el 30% de rendiment, només utilitzant un 5% més de recursos.

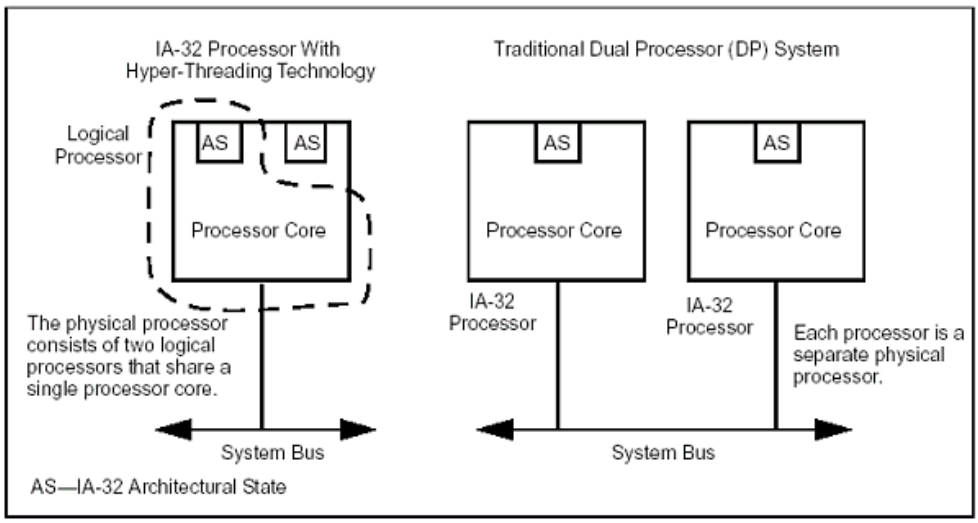

Figura 4.3 Hyperthreading

En l'anterior figura es pot veure la diferencia entre un processador amb dos nuclis físics i un processador amb un nucli físic però que mitjançant l'hyperthreading té dos nuclis lògics.
Un dels processadors que podem trobar en el mercat que incorpori aquesta tecnologia és el Intel Pentium 4 @ 3.80Ghz. Actualment els intel Core i7 també incorporen aquesta tecnologia.

4.2.2 Execució especulativa

Un altres dels mètodes emprats en el disseny de processador per a millorar el seu rendiment és el de l'execució especulativa. Aquesta és l'execució d'un codi per part del processador que a priori no té perquè ser necessària. Només és útil quan la execució prèvia requereix menys temps i espai que el que requeriria l'execució posterior, sent aquest un estalvi el suficientment important com per a compensar el esforç gastat en el cas que el resultat de la operació mai arribi a utilitzar-se. Normalment aquesta tècnica s'utilitza per a reduir el cost computacional de les instruccions de salt condicional, on el processador intenta encertar on és més probable que salti (predicció de salts) i immediatament comença a executar el codi que comença en aquella area. Si posteriorment es demostra que la predicció ha sigut dolenta, tot lo que s'ha executat després del salt es descarta. Aquesta execució prematura és rentable en termes d'ús de recursos doncs si no el pipeline es pararia fins a conèixer la pròxima instrucció a executar-se després del salt. El primers processadors en incorporar aquesta tècnica van ser els Pentium III sorgits el Març de 1999.

4.2.3 Ordenació d'instruccions i reanomenament de variables

Aquest mètode es utilitzat per evitar el conflicte de dependències entre instruccions donat que en un programa poden haver dues instruccions consecutives en la qual la segona depengui de la dada que es generi a la primera, provocant una alentiment a la hora d'executar el pipeline. El que es pretén amb aquest mètode es tornar a ordenar les instruccions i renombrar les variables de manera que es creïn les menors dependencies

possibles, fent així que el processador s'executi el més ràpidament possible. Un exemple d'aquest mètode seria el següent:

Instrucció 1: R1= R2 + 3 → R1 = R2 + 3
Instrucció 2: R1 = 4 + 6 → R3 = 4 + 6

Com podem veure, si el processador tingues dues unitats de càlcul, tardaria 2 cicles a fer les operacions que teníem, donat que les dues es guarden a R1 i per tant un cop realitzada la primera, després es realitzaria la segona. Mentre que mitjançant el renombrament de variables, ara es pot fer les dues operacions a la vegada donat que que la segona instrucció es passa a guardar a R3.

4.3 Arquitectura de computadors

Un cop vistes dues de les diferents arquitectures de memòria que existeixen i diferents mètodes de millora de rendiment d'un processador, es procedirà a explicar en que consisteix els segmentat d'un processador i quines arquitectures diferents existeixen. Primerament dir que la segmentació és un mètode mitjançant el qual s'aconsegueix augmentar el rendiment d'alguns sistemes electrònics digitals. Existeixen varies tècniques. La primera que es veurà és el pipeline.

4.3.1 Aquitectura Pipeline

Aquesta arquitectura consisteix en segmentar un programa mitjançant diverses tuberies fent que es millori el temps d'execució d'aquest. Sabem que cada instrucció té varies etapes, com per exemple l'etapa IF, DEC, EXE. El que s'aconsegueix amb l'arquitectura pipeline és que mentre per una tuberia s'està realitzant l'etapa del EXE, per un altre ja s'estigui descodificant la següent instrucció que s'executarà. Una possible execució de 4 instruccions amb un processador pipeline seria la següent:

Etapes/Cicles	1	2	3	4	5	6
EXE			I1	I2	I3	I4
DEC		I1	I2	I3	I4	
IF	I1	I2	I3	I4		

Taula 4.1 Pipeline

Podem veure que les 4 instruccions només tarden 6 cicles en acabar d'executar-se, mentre que en un processador normal tardarien 12 cicles:

Etapes/Cicles	1	2	3	4	5	6	7	8	9	10	11	12
EXE			I1			I2			I3			I4
DEC		I1			I2			I3			I4	
IF	I1			I2			I3			I4		

Taula 4.2 Procesador sense pipeline

Per tant es pot comprovar que amb una arquitectura pipeline es millora considerablement el rendiment d'un processador.

4.3.2 Arquitectura Superescalar

L'arquitectura superescalar utilitza el paral·lelisme d'instruccions a més a més del paral·lelisme de flux, aquest últim gràcies a l'estructura pipeline vista anteriorment. L'arquitectura superescalar consta d'un pipeline amb les següents etapes:

- Lectura (*fetch*).
- Descodificació (*decode*).
- Llançament (*dispatch*).
- Execució (*execute*).
- Escriptura (*writeback*).
- Finalització (*retirement*).

Aquest processador executa més d'una instrucció cada etapa. El número màxim d'instruccions que por realitzar el processador en una etapa concreta ve determinada per el grau. Així un Processador de grau 2 en lectura (fetch) executaria l'anterior programa de la següent manera:

Etapes/Cicles	1	2	3	4
EXE			I1/I2	I3/I4
DEC		I1/I2	I3/I4	
IF	I1/I2	I3/I4		

Taula 4.3 Arquitectura superescalar

Com es pot apreciar, el temps d'execució de les 4 instruccions a passar a ser de 6 cicles a 4 cicles. El grau de l'etapa d'execució depèn del número i del tipus de les unitats funcionals.

L'arquitectura superescalar va apareixer per primera vegada l'any 1965 en la CDC 6600 de Seymour Cray. A partir de 1998 la gran majoria de CPU dissenyades són superescalars.

4.3.3 Arquitectura VLIW

Els processadors amb arquitectures VLIW es caracteritzen per tenir jocs d'instruccions molt simples en quant a número d'instruccions diferents, però molt grans en quant al tamany de cada instrucció. Això és degut a que cada instrucció especifica el estat de totes i cada una de les unitats funcionals del sistema, amb l'objectiu de simplificar el disseny del hardware i deixar tot el treball de planificar el codi en mans del programador / compilador. Com podem veure es diferencia de l'arquitectura superescalar en que aquesta arquitectura el codi el planifica el compilador, mentre que en l'arquitectura superescalar és el harware el que planifica les instruccions.

Les avantatges que té aquesta arquitectura envers les vistes anteriorment és que aquesta arquitectura és més simple en quan a hardware i a més a més redueix la potència i el consum. Per altra banda, aquesta arquitectura requereix compiladors molt més complexos i alhora qualsevol millora de l'arquitectura requereix canvis en el joc d'instruccions.

Un exemple de com seria una instrucció en l'arquitectura VLIW és el següent:

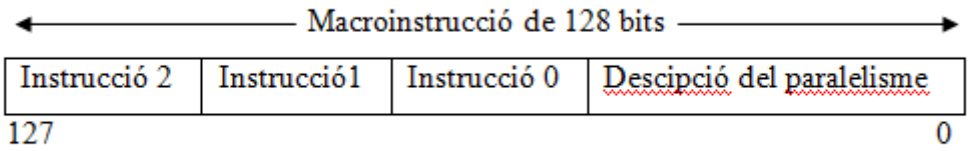

Figura 4.4 Arquitectura VLIW

Es pot veure que el tamany de la instrucció és molt gran, però així es pot especificar quina instrucció ha d'executar cada unitat funcional, evitant que ho hagi de fer el compilador.

Un dels processadors típics que utilitzen aquesta arquitectura és el IA-64 (Intel Itanium), que utilitza 64 bits i va ser dissenyada en cooperació amb Hewlett-Packard.

5 Introducció al LittleProc

El LittleProc és un processador simple que va ser dissenyat per a utilitzar en l'àmbit docent per una sèrie de professors de la UAB.
Aquest processador va ser dissenyat en l'arquitectura Von Neumann. Per tant té una memòria compartida de dades i d'instruccions externa a la UP i a la UC. En la següent figura es veurà com amb quins blocs està dissenyat el LittleProc.

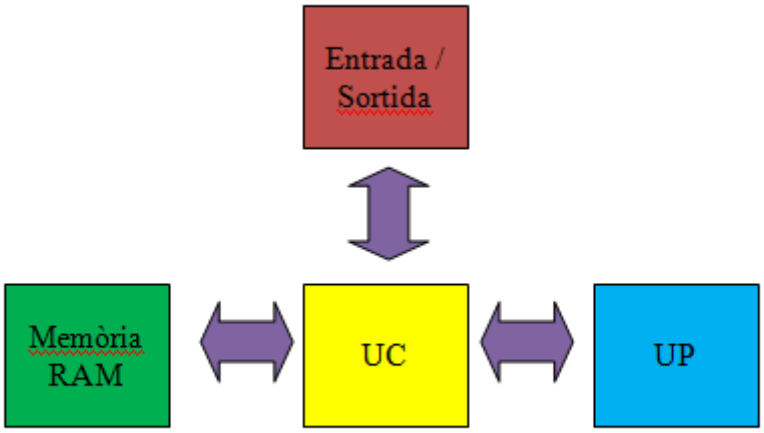

Figura 5.1 LittleProc

Tal i cos es pot veure i com s'ha comentat anteriorment el LittleProc tindrà tres components ben diferenciats l'un de l'altre com són la memòria, la UC i la UP. Aquest processador és de 12 bits, i té un instruction set de 10 instruccions. Algunes d'aquestes instruccions que pot realitzar són el BRE i JMP, Load R1 i R2, ADD, SUB... Seguidament s'enumerarà les principals característiques d'aquest processador:

- Processador de 12 bits.
- Freqüència de rellotge de 10 MHz.
- Arquitectura Von Neumann basat amb el model de 2 busos.
- Memòria de Dades / Programa de 128 Words escalable a 4096 Words.
- Memòria ROM interna de 64 Words.
- Unitat de procés seqüencialitzada.

- Instruction set de 10 instruccions.

5.1 UP

L'unitat de procés és la que s'encarregarà d'emmagatzemar el conjunt de registres que s'encarreguen del correcte funcionament de la execució d'instruccions, així com de realitzar les operacions necessàries. L'esquema que segueix la UP del LittleProc és el següent:

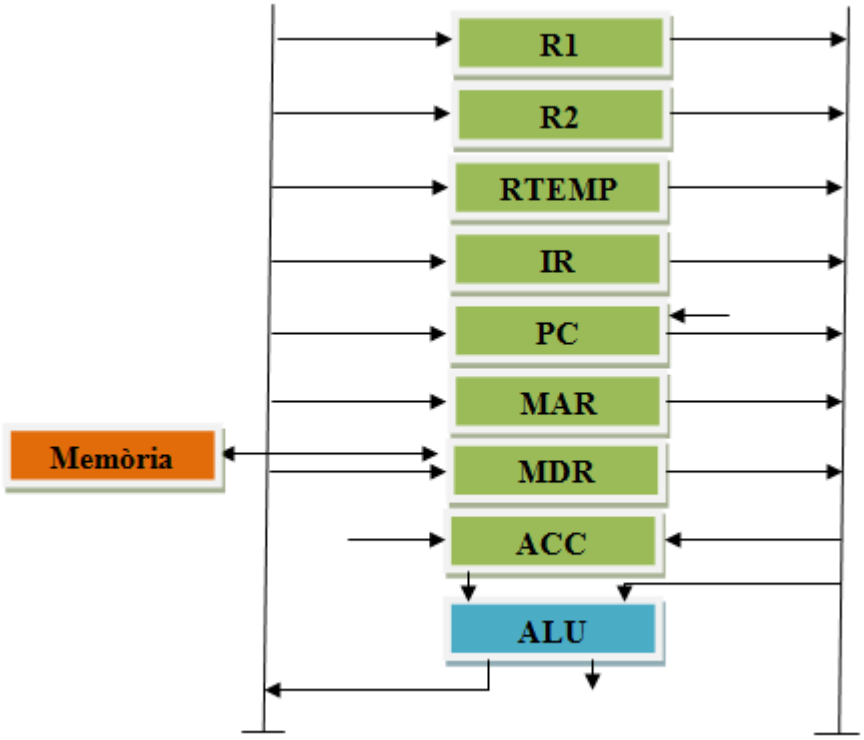

Figura 5.2 UP LittleProc

Com podem veure, el LittleProc té 3 registres a on podrem guardar dades portades de memòria o resultants de les operacions realitzades en la ALU. A més d'aquests tres registres, també existeix el registre IR. Aquest és l'encarregat d'emmagatzemar la instrucció a ésser executada a continuació. Un altre dels registres que podem veure que hi ha és el PC, que s'encarrega d'emmagatzemar l'adreça física de la següent instrucció a ésser executada. Aquesta adreça correspon a la posició de memòria de la RAM. Un alter registre que veiem és el MAR, que s'encarrega d'emmagatzemar l'adreça de la posició de memòria on es troba una dada que s'ha de recollir. El MR és un

altre registre encarregat d'emmagatzemar en aquest cas o bé el contingut d'una adreça d'una posició de memòria interna o bé una dada que procedeix pel bus 2 (ALU). Finalment, l'últim dels registres que ens trobem és el ACC, aquest és l'encarregat d'emmagatzemar temporalment la dada d'un operand de l'ALU. Aquest últim registre és necessari per tal de poder entregar a l'ALU dos operand al mateix temps, donat que degut a l'arquitectura que s'ha utilitzat només té dos busos, només un d'ells ens serveix com a entrada de l'ALU.

L'últim component que veiem que té la UP és la unitat aritmètica lògica (ALU). Aquesta és l'encarregada de realitzar totes les operacions necessàries per al correcte funcionament del processador. La ALU del LittleProc pot realitzar les següents operacions:

- Pas transparent de dades
- Suma
- Resta
- Màscara dels 4 primers bits
- Incrementador d'una unitat
- Desplaçament a la dreta B vegades
- Desplaçament a l'esquerra B vegades

5.2 UC

La unitat de control és l'encarregada de generar la seqüència d'ordres necessàries per a executar cada instrucció emmagatzemada en la memòria. Les instruccions tenen un tamany de 12 bits on es poden diferenciar dos parts:

- Part alta de la instrucció [11...8] també anomenada opcode. Aquests 4 bits són els encarregats de codificar la instrucció a executar. D'aquesta manera amb 4 bits es poden codificar fins a 16 instruccions (2^4= 16).
- A la part baixa de la instrucció [7...0] i trobem 8 bits que poden tenir diferents significats segons l'opcode que l'acompanya:

- Dada numèrica de 8 bits.
- Adreça de 8 bits on es troba la dada a processar dins de la memòria.
- Adreça de salt en cas d'un salt condicional o incondicional.

El procés que segueix la UC per a executar les instruccions consta bàsicament de dos parts, el Fetch i l'execució. Per a cada instrucció el Fetch dura 5 cicles i sempre és el següent:

1. EXE (MAR ← PC)
2. EXE (NOOP)
3. EXE (MDR MEM (MAR))
   EXE (PC ++)
4. EXE (IR ← MDR)
5. BRA

Mentre que l'execució depèn de la instrucció que es vol dur a terme en cada cas.
Els elements que formen la unitat de control són els següents:

- Un seqüenciador que té per objectiu entregar a la seva sortida l'adreça de la següent microinstrucció a executar. Està format bàsicament per 3 elements. El primer d'ells és l'incrementador, que incrementa en 1 el valor introduït a la seva entrada. També hi ha un registre anomenat micropc que serveix per a guardar el número de microinstrucció que s'està executant en cada moment. Finalment dintre del seqüenciador hi podem trobar un multiplexor encarregat de seleccionar quina entrada s'agafarà.
- Una ROM. Aquesta és una memòria de 64 posicions amb una amplada de 25 bits que contindrà totes les microinstruccions de cada instrucció que té el instruction set del processador.
- Un multiplexor de condicions d'entrada. Aquest multiplexor tindrà dues entrades i ens permetrà seleccionar entre l'entrada inici, que quan valgui 1 es començarà a executar tot el programa, i l'entrada

Bit zero, que val 1 quan el resultat d'una operació de la ALU val o i així es pot utilitzar per a realitzar els salts condicionals.

- Decodificador d'instruccions que és l'encarregat de decodificar el opcode i convertir-lo en una adreça real de memòria per tal d'identificar adequadament on comença el seu cicle d'execució dins de la ROM.

5.3 Memòria Principal (RAM)

Com s'ha explicat anteriorment, el LittleProc està dissenyat amb una arquitectura Von Neumann, utilitzant només una memòria per a emmagatzemar-hi les dades i que alhora conté les instruccions de cada programa a executar per el processador. Aquesta memòria és de 128 posicions amb una amplada de 12 bits.
La RAM està dissenyada amb 4 entrades. La primera d'elles és la senyal de rellotge, que estarà connectat al rellotge genèric però invertit. Després tenim l'entrada del bus de dades, per on és llegiran les dades que després s'han d'emmagatzemar. Un altra entrada és el bus d'adreces, connectat al MAR per a poder llegir l'adreça de la instrucció que s'ha d'executar en cada moment. I finalment té l'entrada wren, que permet seleccionar si el que es vol és llegir o si per altre banda és vol escriure una dada.

5.4 Interfície del processador

El LittleProc té la següent interficie amb l'exterior:

- Entrades:
- Ck: és l'entrada de rellotge i té una freqüencia de 10 MHz
- Inicio: és l'entrada que permet possar en funcionament el processador, actiu a alta.
- glResetn: és la entrada que ens permet reiniciar el processador, actiu a baixa.

- Sortides:

- End : S'activa quan el processador a acabat l'execució d'un programa.
-

Tot seguit es pot observar el símbol del LittleProc:

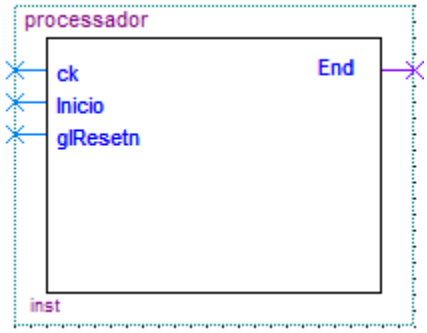

Figura 5.3 LittleProc 1.0

5.5 Funcionament general

Tot seguit s'explicarà quin és el funcionament general del LittleProc 1.0. Aquest, com ja sabem, s'inicia quan rep una senyal externa d'inici, mentre que si rep una senyal de reset es reinicia. Un cop s'ha iniciat el processador, es procedeix a dur a terme l'execució d'un determinat programa carregat a la memòria RAM. El registre PC, el primer cop el tenim a valor 0, i per tant, mitjançant el FETCH, es busca quina és la primera instrucció a executar. Un cop carregat al registre MAR el valor del PC, s'incrementa el PC en 1 per així poder executar la següent instrucció del programa. Un cop realitzada tota la fase del FETCH, es procedeix a executar la instrucció que pertoqui. Per exemple si és una suma, el que es fa és mitjançant la unitat aritmètica lògica de la unitat de procés, sumar els valors dels registres R1 i R2 que prèviament han hagut d'estar carregats mitjançant dues instruccions prèvies i es guarda el resultat al registre RTemp. Un cop executada aquesta instrucció, es va a la primera microinstrucció de la fase FETCH mitjançant un GOTO 2 i es torna a fer la cerca de la següent instrucció a executar. Així es fa fent successivament durant tot el programa fins que al final, un cop s'ha executat tot, s'activa la senyal de END del LittleProc.

6 LittleProc: Avanç cap als 16 bits

La primera modificació realitzada al LittleProc ha sigut la d'augmentar de 12 als 22 bits les instruccions. D'aquests 22 bits, els 6 bits més significatius [21...16], són destinats al opcode, podent codificar un total de 64 instruccions diferents (2^6 = 64), mentre que els altres 16 bits menys significatius estan dedicats a les dades, com poden ser dades numèriques, adreces de salt ...

Per altra banda, també s'ha augmentat el número de registres de la unitat de procés. Tenint ara a part del registre 1, registre 2 i el registre temporal sis registres més, anomenats registre 3, registre 4, registre X, registre Y, registre SP i registre Port1. Al annex 1, es pot veure com a quedat la UP.

En quant a la UC de control s'ha tingut de modificar la memòria ROM de microinstruccions, donat que ara es necessiten un total de 40 senyals de control, i per tant, les microinstruccions de la ROM ara tenen una amplada de 43 bits. D'aquests 43 bits, 38 són per a controlar totes les senyals de la unitat de procés. Dels altres 5 restants, 2 ens indiquen quin tipus de instrucció és, i els altres tres ens permeten seleccionar amb quina condició es saltarà en cas de que sigui un salt, com per exemple, si és negatiu, si és zero, o bé si hi ha hagut una interrupció... Cal recordar que en el LittleProc només hi havia la condició de salt en el cas que fos zero. A més a més també podrem controlar si s'ha produït una interrupció o bé si es necessita llegir o escriure des de l'exterior del processador a través d'un port d'entrada/sortida.

En aquesta modificació del LittleProc, també s'ha ampliat el nombre d'instruccions que conté el instruction set, passant a realitzar-ne 48 instruccions. ***Nota:*** *El instruction set es pot veure al annex 1.*

Anomenar també que en aquesta primera evolució del LittleProc 1.0, s'ha afegit un control de si es produeix alguna interrupció, poder executar-la i

posteriorment retornar a l'execució del programa que s'estava duent a terme.

A més a més també s'ha introduït el PORT1, que és un registre que ens permet llegir i escriure dades des de/ a l'exterior del nostre processador, sent la porta d'entrada/sortida d'aquest.

Seguidament s'enumeraran les principals característiques d'aquest processador comparades amb el LittleProc:

- Processador de 16 bits, mentre que el LittleProc en tenia 12.
- Freqüència de rellotge de 10 MHz tal i com tenia el LittleProc.
- Arquitectura Von Neumann basat amb el model de 2 busos.
- Memòria de Dades / Programa de 128 Words escalable a 4096 Words.
- Memòria ROM interna de 256 Words, mentre que el LittleProc tenia 64 Words.
- Unitat de procés seqüencialitzada.
- Instruction set de 48 instruccions, mentre que el LittleProc en tenia 13.
- Opcode de 6 bits per tant poden codificar 64 operacions, mentre que el LittleProc el tenia de 4 bits poden codificar només 16 operacions.
- Es necessiten 43 bits de senyals de control en la UC, en canvi, en el LittleProc només es tenien 25 bits de control. També es pot observar que en aquesta millora del LittleProc, la unitat de procés passa a necessitar 38 bits de control, mentre que anteriorment només es necessitava 22 bits.
- En quant a la ALU, ara realitzarà les operacions de dividir, multiplicar, decrementar en 1, una màscara e 16 bits i les operacions lògiques AND, OR, XOR, NAND, NOR, i NOT que abans no es realitzaven. Al augmentar el número d'instruccions que pot realitzar la ALU s'ha tingut d'augmentar el número de bits per a seleccionar cada operació, passant de 3 a 5 bits.

- Introducció del control de les interrupcions, donat que el LittleProc no controlava si es produïa una interrupció.
- Introducció d'un registre anomenat Port1 que ens serveix per a llegir o escriure dades cap a l'exterior del processador

Per veure més detalladament el LittleProc de 16 bits, es pot consultar l'annex 1.

7 Segmentació del LittleProc

Com ja s'ha comentat anteriorment, unes de les arquitectures que aconsegueixen millorar el rendiment d'un processador és la segmentació de d'aquest. El que s'ha fet és segmentar el LittleProc, permetent que al mateix temps que s'està executant una instrucció, ja es realitzi la fase del fetch de la següent instrucció o part de ella, arribant en algun cas a poder executar una instrucció immediatament després de que acabi d'executar-se l'anterior. Per a poder realitzar aquesta implementació, s'ha afegit una unitat aritmètica lògica més, i així per exemple mentre es realitza una suma en una unitat de càlcul, amb l'altre es pot calcular el program counter de la següent instrucció a executar. Per altra banda dir que ja que en aquest cas la UC i la UP es compliquen bastant, degut a que hi ha moltes senyals que controlar. Per tant en aquest cas no es controla si es produeix una interrupció.

Per a poder realitzar la segmentació, s'han afegit doblat els dos busos que tenia la UP, ja que dos busos funcionen amb la primera ALU i els dos altres funcionen amb la segona. Dir també que s'han doblat les sortides de cada registre per així poder utilitzar les dades o bé amb una ALU o bé amb l'altre. I a l'entrada de cada registre també s'ha afegit un multiplexor per així poder controlar de quin bus s'agafa la dada. El instruccion set d'aquesta implementació té una instrucció més que l'anterior. Per una banda s'ha eliminat la instrucció de RET que era la que retornava quan s'havia produït una interrupció, ja que en aquest cas no es controlen. Però per altre banda s'ha afegit dos instruccions que ens permeten moure una dada des de el RTemp al R3 o R4.

En quant a la UC en aquest cas es necessita que tingui 56 bits, i que d'aquests es necessitin 52 per a controlar la UP.

També s'ha de comentar que en aquesta implementació s'utilitza l'arquitectura Harvard en quan a memòria es refereix, utilitzant una memòria específica per les instruccions i una diferent per a les dades.
Seguidament es poden veure les principals característiques d'aquest processador comparades amb el LittleProc original:

- Processador de 16 bits, mentre que el LittleProc en tenia 12.
- Freqüència de rellotge de 10 MHz tal i com tenia el LittleProc.
- Arquitectura Harvard amb 4 busos a la UP, mentre que en el LittleProc s'utilitza l'arquitectura Von Neumann basada amb el model de 2 busos.
- Hi ha una memòria de dades independent de la memòria d'instruccions mentre que el LittlePRoc tenia una memòria de Dades / Programa de 128 Words escalable a 4096 Words.
- Memòria ROM interna de 256 Words, mentre que el LittleProc tenia 64 Words.
- Unitat de procés seqüencialitzada.
- Instruction set de 46 instruccions, mentre que el LittleProc en tenia 13.
- Opcode de 6 bits per tant poden codificar 64 operacions, mentre que el LittleProc el tenia de 4 bits poden codificar només 16 operacions.
- Es necessiten 55 bits de senyals de control en la UC, en canvi, en el LittleProc només es tenien 25 bits de control. També es pot observar que en aquesta millora del LittleProc, la unitat de procés passa a necessitar 52 bits de control, mentre que anteriorment només es necessitava 22 bits.
- En quant a la ALU, dir que ara existiran dues ALUs que realitzaran les operacions de dividir, multiplicar, decrementar en 1, una màscara e 16 bits i les operacions lògiques AND, OR, XOR, NAND, NOR, i NOT que abans no es realitzaven. Al augmentar el número d'instruccions que pot realitzar la ALU s'ha tingut d'augmentar el número de bits per a seleccionar cada operació, passant de 3 a 5 bits.

- Segmentació de les instruccions produint així una millora substancial en el rendiment, arribant a executar-se una instrucció immediatament després de que acabi l'anterior.

Per veure més detalladament el LittleProc de 16 bits segmentat, es pot consultar l'annex 2.

8 El LittleProc superescalar: Multi-core de dos processadors

La versió definitiva que s'ha implementat a sigut la versió multicore de dos processadors LittleProc segmentats de 16 bits cada un. Per a crear aquesta implementació el que s'ha fet és unir dos processadors a la mateixa memòria de dades i a la mateixa memòria d'instruccions. Fent això, es poden executar dues instruccions al mateix temps, una en cada processador, augmentant considerablement el rendiment d'aquest processador. Per fer-ho, el que s'ha fet és tal i com s'ha dit unir cada un dels processadors a la memòria de dades i a la memòria d'instruccions. Per a poder interactuar amb els dos processadors a la vegada, s'ha tingut de fer que aquestes memòries fossin de doble port d'entrada i de doble port de sortida tal i com es pot observar en la següent figura.

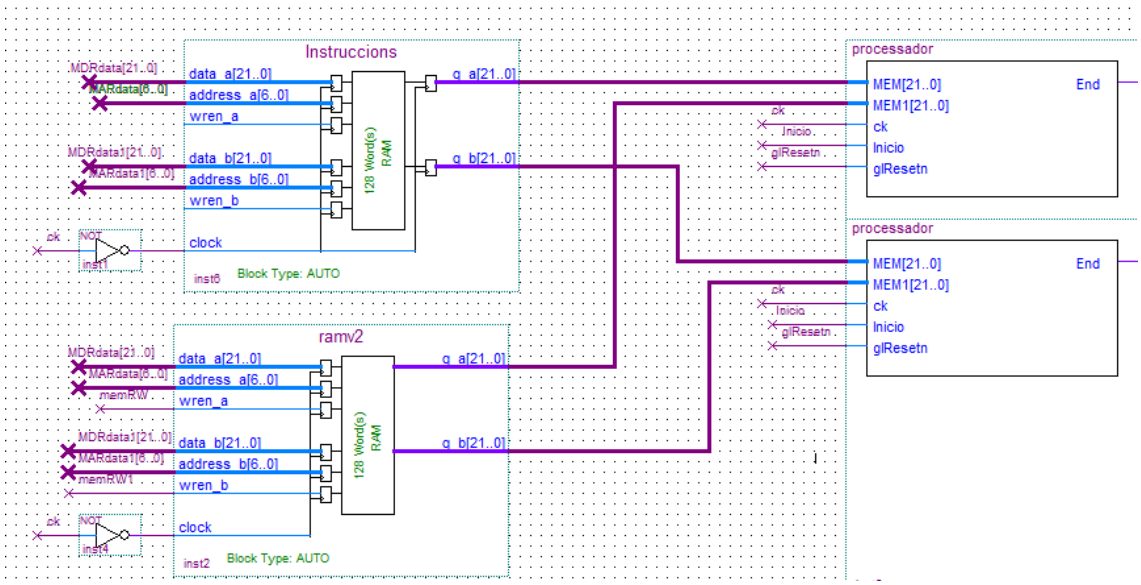

Figura 8.1 LittleProc de dos nuclis

Tal i com es pot veure, a la memòria d'instruccions hi ha dos ports d'entrada i dos de sortida, un que agafa o envia les instruccions al primer processador i un altre que les agafa o les envia al segon processador. Per la memòria de dades passa exactament el mateix, tenint cada processador una entrada per cada memòria respectivament. Això passa perquè s'utilitza

l'arquitectura de Harvard en la memòria. Si s'utilitzés l'arquitectura Von Neumann cada processador només tindria una entrada per la memòria.
Pel que respecta a cada processador, es pot dir que internament és igual que el LittleProc segmentat, només afegint tres instruccions al instruction set de cada un per tal de poder realitzar la partició d'instruccions que ha d'executar cada processador. Aquestes tres instruccions noves estan explicades detalladament al annex 3. Seguidament s'enumeren les característiques de cada un dels dos processadors respecte el LittleProc original.

- Processadors de 16 bits, mentre que el LittleProc en tenia 12.
- Freqüència de rellotge de 10 MHz tal i com tenia el LittleProc.
- Arquitectura Harvard amb 4 busos a la UP, mentre que en el LittleProc s'utilitza l'arquitectura Von Neumann basada amb el model de 2 busos.
- Hi ha una memòria de dades independent de la memòria d'instruccions mentre que el LittlePRoc tenia una memòria de Dades / Programa de 128 Words escalable a 4096 Words.
- Memòria ROM interna de 256 Words, mentre que el LittleProc tenia 64 Words.
- Unitats de procés seqüencialitzada.
- Instruction set de 49 instruccions cada processador, mentre que el LittleProc en tenia 13.
- Opcodes de 6 bits per tant poden codificar 64 operacions, mentre que el LittleProc el tenia de 4 bits poden codificar només 16 operacions.
- Es necessiten 55 bits de senyals de control en les UCs, en canvi, en el LittleProc només es tenien 25 bits de control. També es pot observar que en aquesta millora del LittleProc, les unitat de procés passen a necessitar 52 bits de control, mentre que anteriorment només es necessitava 22 bits.
- En quant a les ALU, dir que ara en tenim dues per processador i que ara realitzaran les operacions de dividir, multiplicar, decrementar en 1, una màscara e 16 bits i les operacions lògiques AND, OR, XOR, NAND, NOR, i NOT que abans no es realitzaven. Al augmentar el

número d'instruccions que pot realitzar la ALU s'ha tingut d'augmentar el número de bits per a seleccionar cada operació, passant de 3 a 5 bits.

- Segmentació de les instruccions produint així una millora substancial en el rendiment, arribant a executar-se una instrucció immediatament després de que acabi l'anterior.

Per veure més detalladament el LittleProc de 16 bits superescalar, es pot consultar l'annex 3.

9 Estudi del rendiment de cada arquitectura

Seguidament es durà a terme un estudi del rendiment de cada arquitectura analitzant les diferent millores produïdes en cada una de elles mitjançant l'execució de determinats programes en cada arquitectura. Primerament executarem un programa que consistirà en realitzar una suma de cada component de dos vectors de 8 posicions cada un de ells. Les dades d'aquest vector seran aleatòries. Posteriorment, realitzarem unes divisions amb 6 vectors de 8, 16 i 32 cada un d'ells.

- Suma de dos vectors de 8 posicions cada un:

- Codi del programa per el LittleProc de 16 bits:

0. 000000 → No fas res
1. 260008 → Carrega al registre X el valor 8 (tamany del vector)
2. 270019 → Carrega al registre Y el valor 25 (direcció on comencen les dades)
3. 280000 →Executes la instrucció LDR1 Y + num(en aquest cas 0)
4. 290008 →Executes la instrucció LDR2 Y + num(en aquest cas 8)
5. 020000 → Executes la suma
6. 2A0010 → Executes la instrucció STRT Y + num (en aquest cas 16)
7. 2B0000 → Incrementes el valor de Y
8. 1F0003 → Executes el salt DJNZ X. En el cas que sigui 0 no saltes, mentre que en el cas que sigui diferents, decrementes en 1 el valor de X i vas a la quarta instrucció del programa
9. 090000 → Finalitzes l'execució del programa amb la instrucció Fi

També s'ha de tenir present que en aquesta implementació les dades estan introduïdes a la mateixa RAM que conté les instruccions, i aquestes estan configurades perquè comencin a la posició 25.
Al executar aquest programa podem observar que el processador acaba l'execució al instant 52.86 us aproximadament.

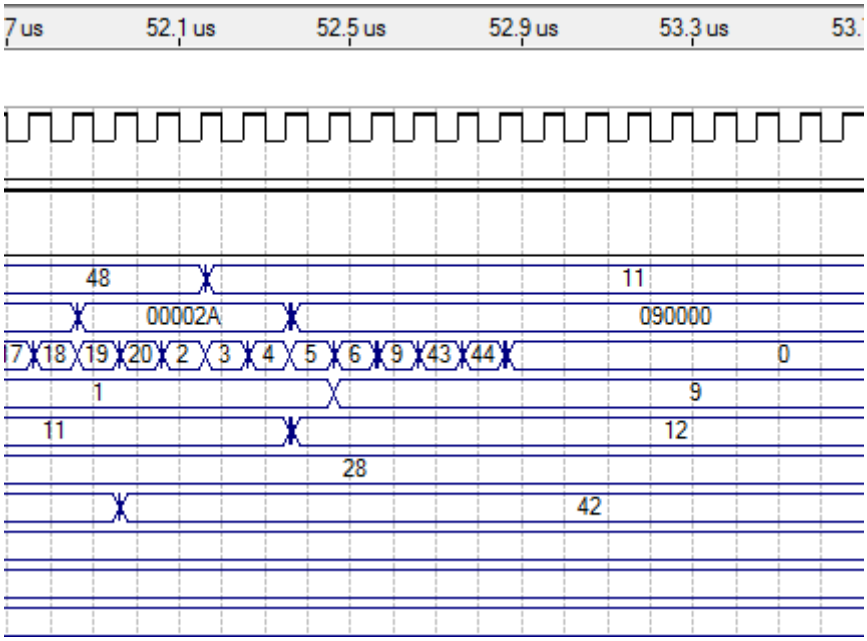

Figura 9.1 Suma de dos vectors de 8 posicions en el LittleProc de 16 bits

- Codi del programa per el LittleProc segmentat:

0. 000000 → No fas res
1. 260008 → Carrega al registre X el valor 8 (tamany del vector)
2. 270000 → Carrega al registre Y el valor 0 (direcció on comencen les dades)
3. 280000 →Executes la instrucció LDR1 Y + num(en aquest cas 0)
4. 290008 →Executes la instrucció LDR2 Y + num(en aquest cas 8)
5. 020000 → Executes la suma
6. 2A0010 → Executes la instrucció STRT Y + num (en aquest cas 16)
7. 2B0000 → Incrementes el valor de Y
8. 1F0003 → Executes el salt DJNZ X. En el cas que sigui 0 no saltes, mentre que en el cas que sigui diferents, decrementes en 1 el valor de X i vas a la quarta instrucció del programa
9. 090000 → Finalitzes l'execució del programa amb la instrucció Fi

Com es pot observar, com que ara utilitzem una memòria de dades independent de la d'instruccions, podem començar a tenir les dades a la posició 0 de la memòria.
Observem com queda la simulació i veiem que acaba en el instant 39.16 us aproximadament.

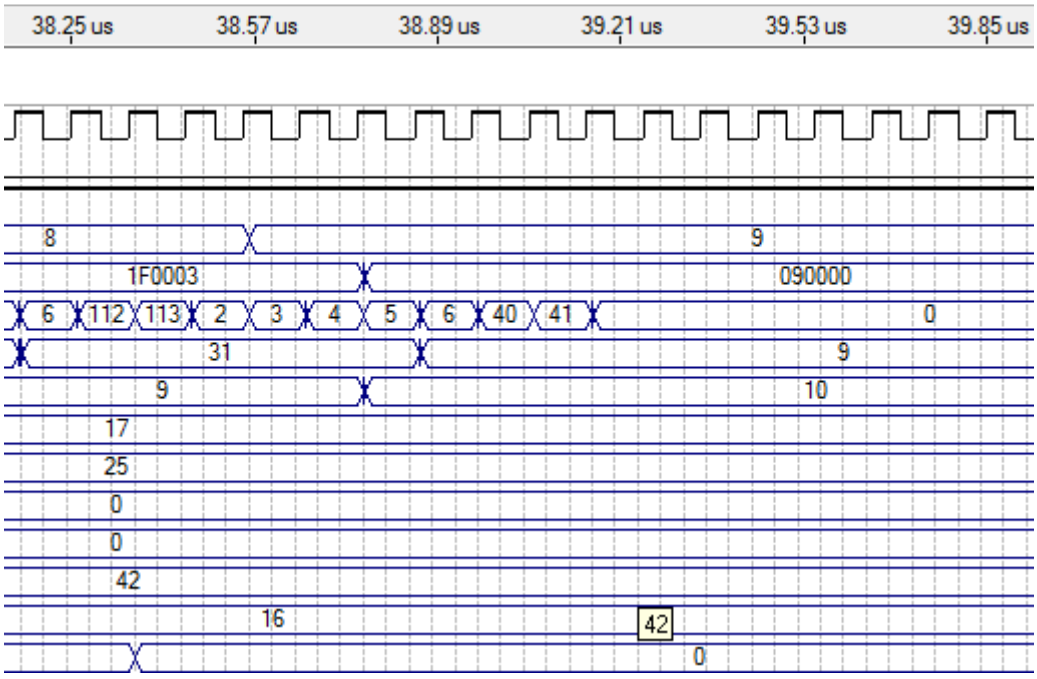

Figura 9.2 Suma de dos vectors de 8 posicions en el LittleProc segmentat

- Codi del programa per el LittleProc superescalar:

0. 000000 → No fas res
1. 250001 → Carregues a R4 el valor de la posició de memòria 1
2. 240000 → Carregues a R3 el valor de la posició de memòria 0
3. 2E0000 → Divideixes els vectors en 2
4. 300000 → Carregues a R4 el doble del tamany dels vectors
5. 270002 → Carrega al registre Y el valor 2 (direcció on comencen les dades) o bé pel segon processador carregues el valor 2 + X
6. 280000 →Executes la instrucció LDR1 Y + num(en aquest cas 0)
7. 290000 →Executes la instrucció LDR2 Y + R3
8. 020000 → Executes la suma
9. 2A0000 → Executes la instrucció STRT Y + R4

10. 2B0000 → Incrementes el valor de Y
11. 1F0006 → Executes el salt DJNZ X. En el cas que sigui 0 no saltes, mentre que en el cas que sigui diferents, decrementes en 1 el valor de X i vas a la setena instrucció del programa
12. 090000 → Finalitzes l'execució del programa amb la instrucció Fi

Ara s'ha de tenir present que a la memòria de dades les dos primeres posicions contenen la següent informació i per això les dades comencen a partir de la tercera posició:

0. 000008 → Indica que el vector té un tamany de 8 posicions
1. 000001 → Serveix per a realitzar un desplaçament

La simulació ens queda de la següent manera, on es pot observar que tarda 22.83 a executar-se tot el programa del processador que acaba últim, on es quan considerarem que s'ha acabat l'execució total.

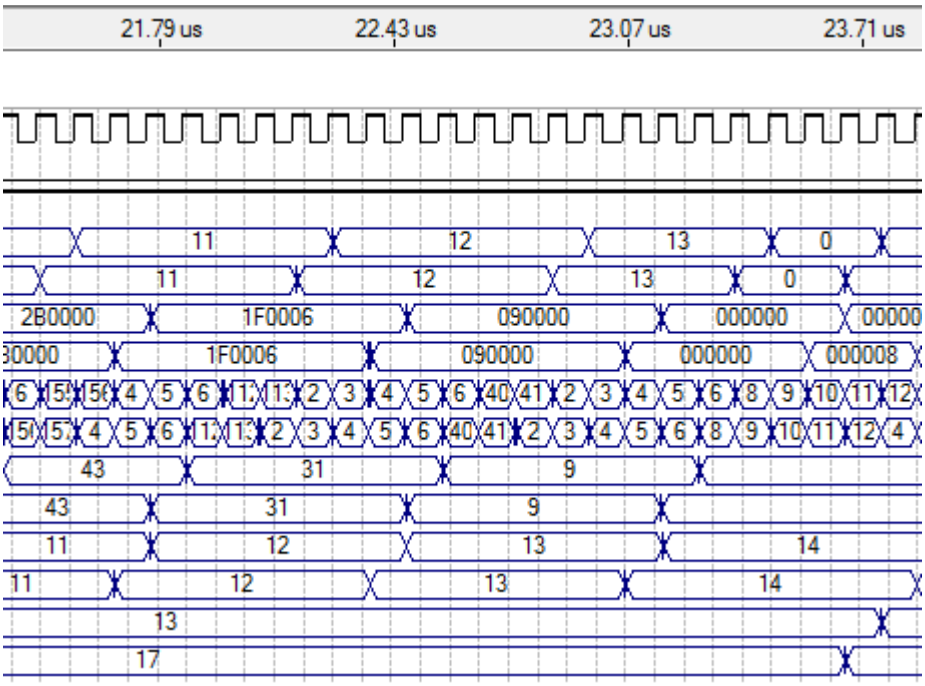

Figura 9.3 Suma de dos vectors de 8 posicions en el LittleProc superescalar

Seguidament es pot veure una taula comparativa dels temps obtinguts en les diferents arquitectures on hem realitzat la mateixa simulació. Tal i com

es pot observar a simple vista el LittleProc superescalar utilitzant dos nuclis millora en més de la meitat el temps d'execució del LittleProc de 16 bits. En la següent taula es pot observar qui és el porcentatge de millora respecte del LittleProc de 16 bits de les altres arquitectures.

Arquitectura	Temps	% Millora
LittleProc de 16 bits	52.86 us	----
LittleProc segmentat	39.16 us	25.92 %
LittleProc superescalar	22.83 us	56.82 %

Taula 9.1 Taula comparativa dels temps d'execució d'una suma de dos vectors de 8 posicions

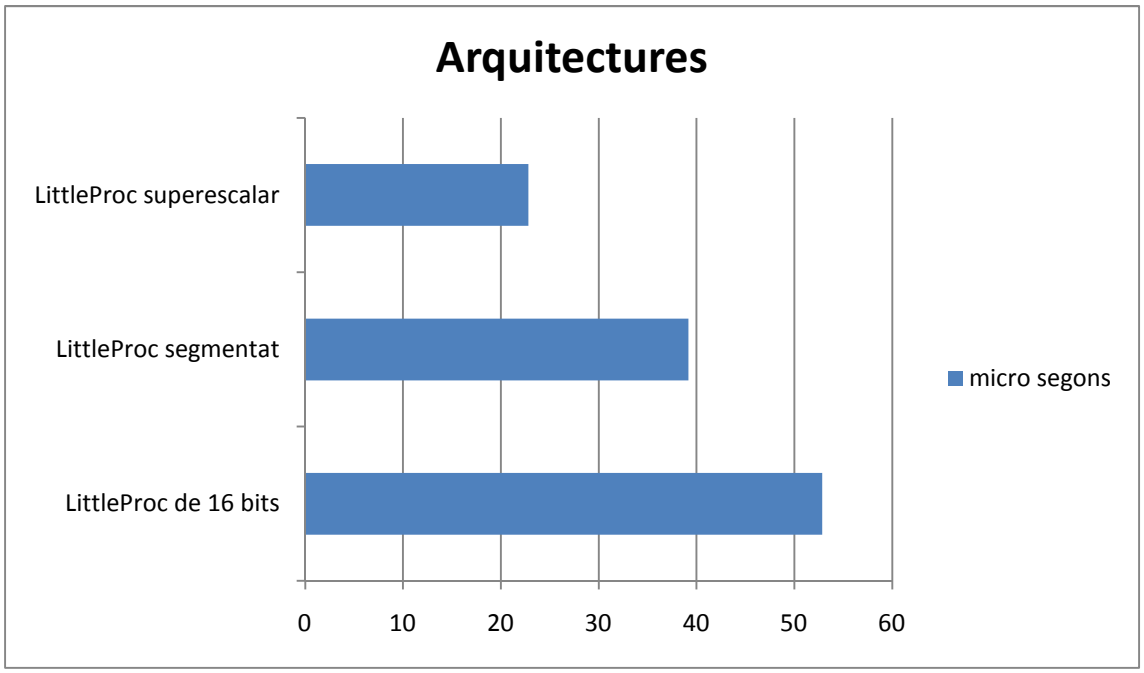

Figura 9.4 Gràfica comparativa del rendiment de les diferents arquitectures en la suma de dos vectors de 8 posicions

Ara anem a dividir dos vectors de 8, 16 i 32 bits respectivament mitjançant les tres arquitectures implementades.

- Codi del programa per el LittleProc de 16 bits:

0. 000000 → No fas res
1. 260008 → Carrega al registre X el valor 8 (tamany del vector)
   260010 → Carrega al registre X el valor 16 (tamany del vector)
   260020 → Carrega al registre X el valor 32 (tamany del vector)
2. 270019 → Carrega al registre Y el valor 25 (direcció on comencen les dades)
3. 280000 →Executes la instrucció LDR1 Y + num(en aquest cas 0)
4. 290008 →Executes la instrucció LDR2 Y + num(en aquest cas 8)
   290010 → Executes la instrucció LDR2 Y + num(en aquest cas 16, pels vectors de 16 bits)
   200020 → Executes la instrucció LDR2 Y + num(en aquest cas 32, pels vectors de 32 bits)
5. 160000 → Executes la divisió
6. 2A0010 → Executes la instrucció STRT Y + num (en aquest cas 16)
   2A0020 → Executes la instrucció STRT Y + num (en aquest cas 32, pels vectors de 16 bits)
   2A0040 → Executes la instrucció STRT Y + num (en aquest cas 64, pels vectors de 32 bits)
7. 2B0000 → Incrementes el valor de Y
8. 1F0003 → Executes el salt DJNZ X. En el cas que sigui 0 no saltes, mentre que en el cas que sigui diferents, decrementes en 1 el valor de X i vas a la quarta instrucció del programa
9. 090000 → Finalitzes l'execució del programa amb la instrucció Fi

També s'ha de tenir present que en aquesta implementació les dades estan introduïdes a la mateixa RAM que conté les instruccions, i aquestes estan configurades perquè comencin a la posició 25.

- Codi del programa per el LittleProc segmentat:

0. 000000 → No fas res
1. 260008 → Carrega al registre X el valor 8 (tamany del vector)
   260010 → Carrega al registre X el valor 16 (tamany del vector)

260020 → Carrega al registre X el valor 32 (tamany del vector)

2. 270000 → Carrega al registre Y el valor 0 (direcció on comencen les dades)
3. 280000 →Executes la instrucció LDR1 Y + num(en aquest cas 0)
4. 290008 →Executes la instrucció LDR2 Y + num(en aquest cas 8)
   290010 →Executes la instrucció LDR2 Y + num(en aquest cas 16, pels vectors de 16 bits)
   290020 →Executes la instrucció LDR2 Y + num(en aquest cas 32, pels vectors de 16 bits)
5. 160000 → Executes la divisió
6. 2A0010 → Executes la instrucció STRT Y + num (en aquest cas 16)
   2A0020 → Executes la instrucció STRT Y + num (en aquest cas 32, pels vectors de 16 bits)
   2A0040 → Executes la instrucció STRT Y + num (en aquest cas 64, pels vectors de 16 bits)
7. 2B0000 → Incrementes el valor de Y
8. 1F0003 → Executes el salt DJNZ X. En el cas que sigui 0 no saltes, mentre que en el cas que sigui diferents, decrementes en 1 el valor de X i vas a la quarta instrucció del programa
9. 090000 → Finalitzes l'execució del programa amb la instrucció Fi

- Codi del programa per el LittleProc superescalar:

0. 000000 → No fas res
1. 250001 → Carregues a R4 el valor de la posició de memòria 1
2. 240000 → Carregues a R3 el valor de la posició de memòria 0
3. 2E0000 → Divideixes els vectors en 2
4. 300000 → Carregues a R4 el doble del tamany dels vectors
5. 270002 → Carrega al registre Y el valor 2 (direcció on comencen les dades) o bé pel segon processador carregues el valor 2 + X
6. 280000 →Executes la instrucció LDR1 Y + num(en aquest cas 0)
7. 290000 →Executes la instrucció LDR2 Y + R3
8. 020000 → Executes la suma
9. 2A0000 → Executes la instrucció STRT Y + R4

10. 2B0000 → Incrementes el valor de Y
11. 1F0006 → Executes el salt DJNZ X. En el cas que sigui 0 no saltes, mentre que en el cas que sigui diferents, decrementes en 1 el valor de X i vas a la setena instrucció del programa
12. 090000 → Finalitzes l'execució del programa amb la instrucció Fi

Ara s'ha de tenir present que a la memòria de dades les dos primeres posicions contenen la següent informació i per això les dades comencen a partir de la tercera posició:

0. 000008 → Indica que el vector té un tamany de 8 posicions
   000010 → Indica que el vector té un tamany de 16 posicions
   000020 → Indica que el vector té un tamany de 32 posicions
1. 000001 → Serveix per a realitzar un desplaçament

Un cop realitzades totes les execucions, ens queda una taula amb els següents temps d'execució.

Arquitectura	Divisió vectors de 8 posicions	Divisió vectors de 16 posicions	Divisió vectors de 32 posicions
LittleProc de 16 bits	52,26 us	101,86 us	201,06 us
LittleProc segmentat	38,34 us	74,36 us	146,35 us
LittleProc superescalar	22,43 us	40,44 us	76,45 us

Taula 9.2 Temps d'execució de les diferents arquitectures en diferents tamanys de vectors

Seguidament es pot veure com queda la comparativa dels temps d'execució entre les diferents arquitectures implementades. Per fer el gràfic s'ha agafat els temps dels tres tamanys diferents de vectors.

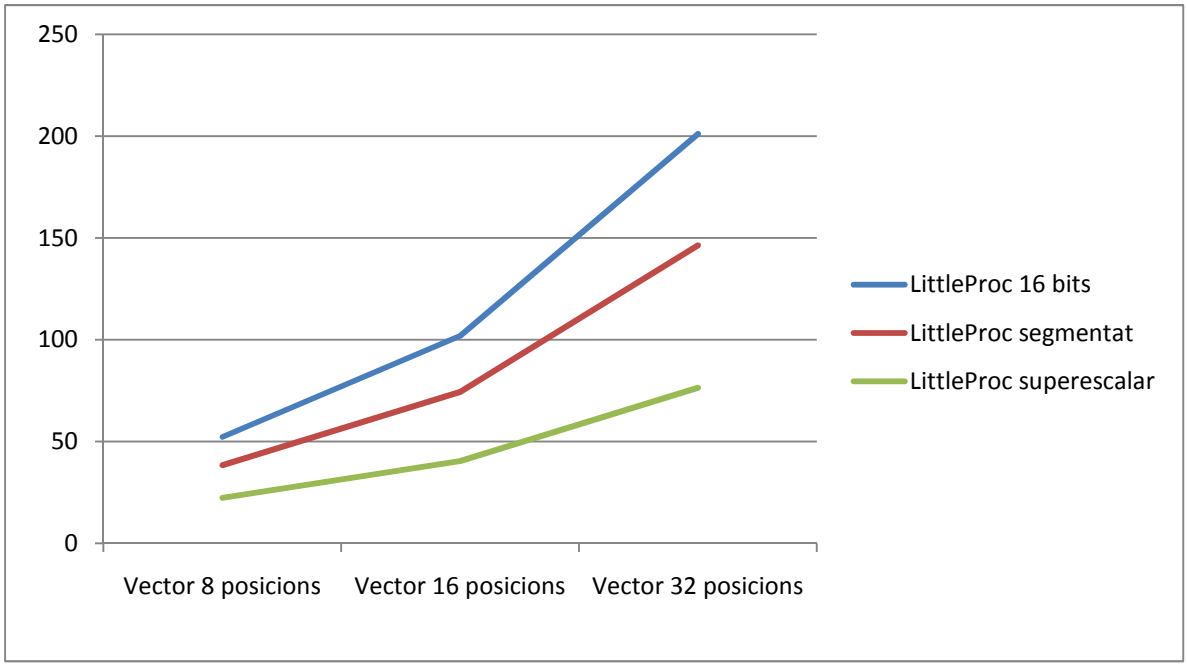

Figura 9.5 comparativa dels temps d'execució per diferents tamanys de vectors

Primerament dir que en el eix Y de l'anterior gràfica, estàn representat els temps d'execució de les tres arqutiectures en microsegons. Mentre que en el eix horitzontal, estan representats els diferents tamanys dels vectors analitzats. Segons es pot apreciar, el LittleProc superescalar te una pendent de creixement del temps d'execució molt menys elevat que el LittleProc de 16 bits, per tant podem deduir que contra més gran sigui el tamany del vector amb el que volem operar, més millora tindrem en el temps d'execució amb el LittleProc superescalar respecte el LittleProc de 16 bits.

9.1 Estudi del temps de CPU

Si realitzem un estudi sobre el temps de CPU sobre el LittleProc superescalar utilitzant la següent fòrmula:

$$T_{CPU} = \text{Nº instruccions} * CPI * T_c$$

I a nosaltres ens interessa saber el CPI per posteriorment poder calcular el número d'instruccions que executa el processador per cicle de rellotge.

Tenim que el temps total d'execució es de 74,45 * 10^{-6}. Aleshores tenim també que el numero d'instruccions és de 32 * 6 (bucle 32 cops) + 7 (instruccions fora del bucle) per tant 199. I també sabem que la freqüència és de 10 MHz.

Per tant si apliquem la formula trobem que:

74,45 * 10^{-6} = 199 * CPI * 10^{-6}

CPI = 74,45 / 199 = 0,3741 cicles / instrucció.

Per tant el nostre processador LittleProc superescalar executarà 1 / 0,3741 = 2,673 instruccions per cicle.

Per altra banda si fem el mateix amb el LittleProc de 16 bits veiem:

201,06 * 10^{-6} = 196 * CPI * 10^{-6}
CPI = 201,06 / 196 = 1,025 cicles / instrucció

1 / 1,025 = 0,9756 instruccions per cicle. Per tant podem observar que no arriba a executar una instrucció per cicle.

9.2 Càlcul d'instruccions per segon

Un cop tenim calculat el número d'instruccions que executa per cicle, podem calcular quin és el nombre d'instruccions que executa per segon. Aquest càlcul és el MIPS i és molt útil per a comparar rendiments de diversos processadors. Realitzarem el càlcul mitjançant la següent fórmula:

T = CPI / període

Si nosaltres sabem que el LittleProc superescalar executa 2,673 instruccions cada cicle, i sabem que el nostre processador treballa a una freqüència de 10 MHz i podem deduir dons que el període serà de 10^{-7} segons. Per tant el nostre processador superescalar executarà 2673000 instruccions cada segon (2,673 MIPs). Comparant amb altres processadors que han existit podem dir que el LittleProc superscalar pot calcular aproximadament el triple d'instruccions cada segon que podia calcular el Intel 8080 (800 KIPs).

10 Conclusió

Durant el desenvolupament del projecte s'han anat assolint tots els objectius marcats al principi d'aquest, a més a més d'incorporar a les diverses versions varis elements nous que no estaven previstos dins del objectius marcats en la planificació del projecte, com poden ser el control de les interrupcions i l'ús d'un port que ens permeti establir una connexió entre el processador i l'exterior.
Respecte al temps establert per a realitzar el processador també es pot considerar que s'ha complert, malgrat els petits retards sorgits per els nous elements incorporats comentats anteriorment.
A nivell de resultats dir que tal i com s'esperava, el rendiment del LittleProc segmentat és més bo que el del LittleProc original. I lògicament degut al ús de dos processadors en un en el superescalar, aquest és el més òptim de tots, arribant a reduir a més de la meitat el temps d'execució respecte el LittleProc 1.0.

10.1 Futures línies de treball

Tal i com s'ha fet amb tot el projecte, cada nova fase implementada és una evolució de l'anterior, sent fàcilment evolucionable gràcies al bon disseny del LittleProc. Per tant respecte a futures evolucions d'aquest processador desenvolupat per mi podrien ser les següents entre d'altres:

- Buscar un nivell més elevat de superescalaritat en el processador multicore mitjançant l'ús de més nuclis d'execució. Comentar que nosaltres utilitzàvem només dos.
- Realitzar un planificador en el mateix multicore que ens permetés dividir l'execució de les instruccions entre els diferents nuclis per a qualsevol programa, degut a que el multicore implementat per mi només permet repartir l'execució d'instruccions sempre i quant el programa que s'executi sigui referent al tractament d'informació contingut en un vector. (Com pot ser calcular la mitjana d'aquest, la suma de tots els components ...)

10.2 Opinió personal

La realització d'aquest projecte a suposat per mi posar en pràctica varis dels coneixements adquirits en gran part de les assignatures estudiades durant tota l'Enginyeria Tècnica en Informàtica de Sistemes com poden ser Sistemes Digitals II, Disseny de Sistemes Digitals, Introducció a l'Arquitectura de Computadors, Estructura de Computadors, Sistemes Operatius ... entre d'altres. I tal i com s'ha comentat en el capítol de la introducció, ha sigut un molt bon projecte per aprofundir en coneixements que posteriorment estudiaré en futurs estudis.

11 Bibliografia

11.1 Llibres

1. Ashenden, Peter J., "The Designer's guide to VHDL", 3er ed, Burlington : Morgan Kaufmann, 2008

2. Brown, Stephen D., "Fundamentals of digital logic with VHDL design", 3er ed, Boston [etc.] : McGraw-Hill, cop. 2009

3. Johnson, Mike, "Superscalar microprocessor desing", Englewood Cliffs : Prentice Hall, cop. 1991

4. Pérez López, Serafín Alfonso, "Diseño de sistemas digitales con VHDL", Madrid : International Thomson, cop. 2002

5. Ruz Ortiz, José Jaime, "VHDL : de la tecnología a la arquitectura de computadores", Madrid : Síntesis, DL 2003

6. Saiz, Joaquín. Portero, Antoni. Aragonès, Raül, "LittleProc: disseny d'un microprocessador en una plataforma reconfigurable", Bellaterra : Universitat Autónoma de Barcelona, 2010

7. Shen, John Paul, "Arquitectura de computadores : [fundamentos de los procesadores superescalares]", Madrid [etc.] : McGraw Hill, DL 2006

8. Omondi, Amos R.,"The Microarchitecture of pipelined and superscalar computers", Boston [etc.] : Kluwer Academic, cop. 1999

9. Uffenbeck, John, "Microcomputers and microprocessors : the 8080, 8085 and Z-80 programming, interfacing and troubleshooting / John Uffenbeck", 2nd ed, Englewood Cliffs : Prentice-Hall International, cop. 1991

11.2 Fonts electròniques

1. " Arquitectura de Von Neumann y arquitectura Harvard" , la tecla de escape [Online], Disponible en: http://latecladeescape.com/basico/arquitectura-de-von-neumann-y-arquitectura-harvard.html (Gener 2011)

2. Pipelinescomplejos, (16/01/2008), "ejecución especulativa" [Online], Disponible en : http://pipelinecomplejos.blogspot.es/ (Gener 2011)

3. "Los principios básicos de arquitecturas de vliw" [Online], Disponible en: http://www.angelfire.com/ca6/angie/vliw.htm (Gener 2011)

4. La wikipedia [online], Disponible en : http://es.wikipedia.org/wiki/Wikipedia:Portada

5. Bárbara García Olivera, Mariuly González Rodríguez, "Arquitectura de ejecución dinàmica en Pentium III" [Online], Disponible en: http://www.monografias.com/trabajos14/pentium/pentium.shtml (Juny 2011)

Llista de taules

Llista de figures