Instituto Tecnológico de Costa Rica

Área Académica de Ingeniería en Computadores

(Computer Engineering Academic Area)

Programa de Licenciatura en Ingeniería en Computadores

(Licentiate Degree Program in Computer Engineering)

Curso: CE-4301 Arquitectura de Computadores I

(Curse: CE-4301 Computer Architecture I)

Informe de Proyecto #1

(Report of project #1)

Realizado por:

Made by:

Nicolás Jiménez García, 201258421

José Daniel Badilla Umaña, 201271708

Profesor:

(Professor)

Juan Manuel Sánchez Corrales

Fecha: Cartago, Septiembre 4, 2016

(Date: Cartago, September 4th, 2016)

Introducción

Los sistemas computaciones han significado un gran progreso para la humanidad por su poder de cálculo, la cual recientemente ha ido en un vertiginoso crecimiento.

Un sistema computacional es bastante complejo y cuenta con varias capas de abstracción, desde los niveles más bajo concernientes con la física de los materiales y su interacción, hasta los niveles más altos de abstracción, que están relacionados con el *software* que se ejecuta en las computadoras.[2]

Una parte importante de estos niveles de abstracción es la arquitectura y la microarquitectura del sistema computacional. Es estas capa en donde se liga el *hardware* con el *software.* Además en esta zona se definen aspectos importantes de cómo va a funcionar las etapas de más alta abstracción y de cómo controlar el *hardware* de la computadora. [3]

En este proyecto se aborda el diseño y la implementación de una arquitectura y micro-arquitectura de un computador. Para realizar esto el proyecto se divide en varias etapas, las cuales son:

Etapa 1: En la primera etapa se define el conjunto de instrucciones (ISA) para un computador RISC (Reduced Instruction Set Computer), esto incluye todos los aspectos que atañen al diseño de la arquitectura del computador, como los son: la clase del ISA, el tamaño de las instrucciones, las operaciones soportadas, la encodificación de las instrucciones, entre otros. En esta etapa se excluye el área de la micro-arquitectura.

Etapa 2: En la segunda etapa se implementa a nivel de simulación por *software* una micro-arquitectura basada en el ISA diseñado en la etapa anterior. Para implementar esta micro-arquitectura se utiliza un lenguaje de programación de alto nivel que modele el comportamiento esperado de una micro-arquitectura física. Cabe destacar que esta implementación se realiza con un modelo escalar, lo que significa que por cada ciclo de reloj, el procesador solo es capaz de ejecutar una instrucción de forma simultánea.

Etapa 3: En la tercera etapa se parte del programa informático de simulación del procesador creado en la etapa anterior, pero se agregan nuevas funcionalidades

que mejoran el rendimiento y velocidad del procesador. Esta micro-arquitectura es de modelo super-escalar lo cual significa que el procesador es capaz de ejecutar múltiples instrucciones por ciclo de reloj. Además de que este procesador utiliza algoritmos de calendarización para poder ejecutar estas instrucciones debidamente.

Marco Teórico

Dado que el proyecto se centraliza en el desarrollo de una arquitectura de computador, se necesita plantear la conceptualización de algunos parámetros y características dados por el mismo para dar un apoyo a la interpretación del cuerpo del informe. Para empezar, tenemos el concepto de ISA, por sus siglas en inglés, es la arquitectura para el set de instrucciones de un computador.[5] Estas son aquellas que van a ser interpretadas por la unidad de procesamiento correspondiente, es decir, las instrucciones serán las encargadas de decirle al procesador que realizar con los datos que se proporcionan.

Cabe recalcar, que el término de arquitectura comprende el mismo significado al aplicarlo en un computador, dicho término, hace referencia al set de instrucciones. Así mismo, el set de instrucciones puede serde dos tipos, CISC o RISC. Cuando se habla de una arquitectura CISC, se tiene que el conjunto de instrucciones es complejo, es decir, que posee un número amplio de instrucciones y además permite operaciones complejas. Mientras tanto, una arquitectura RISC, hace referencia a un set de instrucciones reducido. Esta arquitectura tiene instrucciones de tamaño fijo y presentadas en un menor cantidad de formatos.

Existen distintos tipos de instrucciones de las cuales se puede mencionar las aritméticas, las lógicas, las de control y las de desplazamiento. Las instrucciones aritméticas son aquellas representadas por sumas, restas, multiplicaciones y estas pueden operar números enteros y a sí mismo con números de punto flotante.

Seguidamente, las instrucciones lógicas son operaciones como AND, OR, NOT, XOR, entre otras que los operan bit a bit. Las instrucciones de control son las que realizan saltos condicionales o incondicionales que permiten ir a cualquier instrucción dada, por último, están las instrucciones de desplazamiento las cuales permiten a una cadena de bits desplazar bit e incluir nuevos bits a la derecho o izquierda de la cadena.

Ahora bien, existen también clases de ISA que se necesitan conceptualizar para analizar y comprender las arquitecturas de computadores. Las clases de ISA son las siguientes: Stack, Acumulador, Registro-Memoria y Registro-Registro. La primera clase se refiere a una Pila(Stack), en la cual, los operandos se encuentran dentro de una pila y para realizar las operaciones se utiliza el operando push para insertar un valor, luego para extraer se utiliza el operando pop. Para ejemplificar la clase de ISA Stack se muestra en la figura 1.

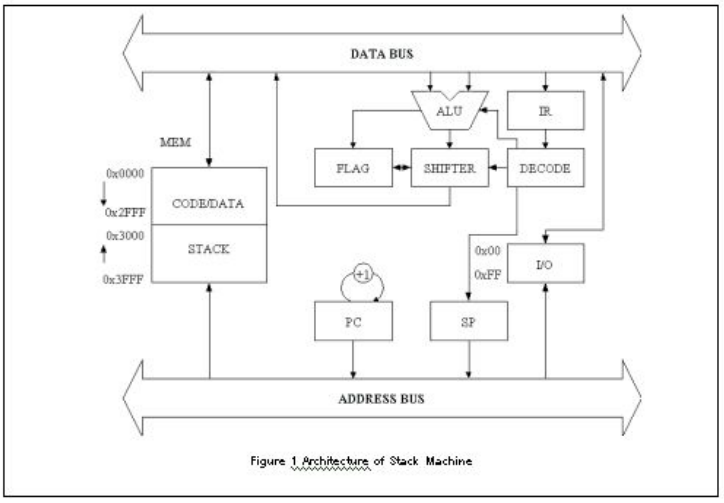

Figura 1. Arquitectura de la clase ISA Stack

Luego se encuentra la clase Acumulador, la cual, se define por la característica de presentar un registro acumulador, se utiliza el operando en el acumulador y el otro se encuentra directamente en memoria. En la figura 2, se puede notar la presencia del registro acumulador que permite a la arquitectura su conformación.

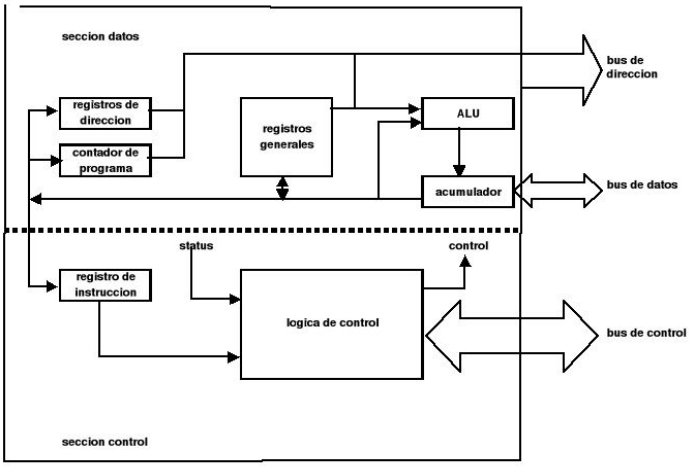

Figura 2. Arquitectura de la clase ISA Acumulador

La clase Registro-Memoria, se define como una arquitectura que cuenta con varios registros de propósito general y una memoria, en el cual, se obtiene un operando de la memoria y otro de los registros. Así también, en la figura 3, se denota el cambio con respecto a las clases de ISA anteriores y se observa la presencia de registro y memoria para realizar las operaciones.

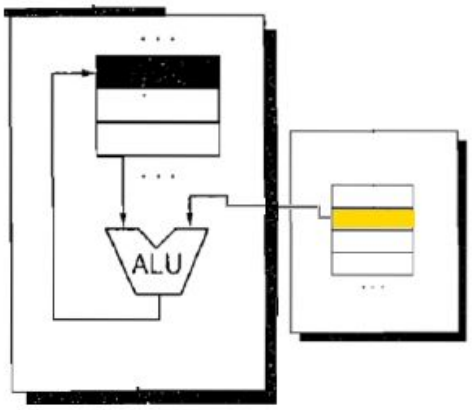

Figura 3. Arquitectura de la clase ISA Registro-Memoria

Por último, la clase Registro-Registro, se asemeja a la anterior con la diferencia que ambos operandos son extraídos de los registros. Viendo la diferencia con respecto a la clase anterior, se ve la aparición del uso de dos registros, como se muestra en la figura 4.

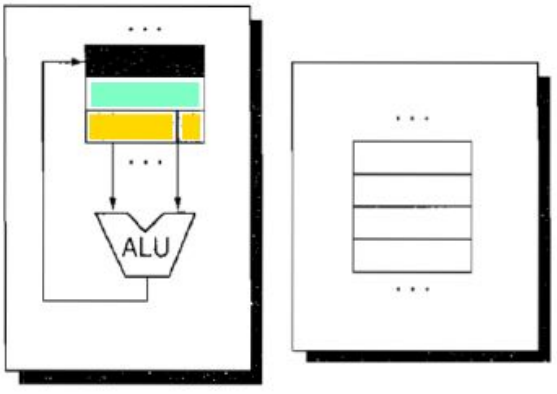

Figura 4. Arquitectura de la clase ISA Registro-Registro

Cabe recalcar que todas las arquitecturas que se mencionaron anteriormente poseen un direccionamiento a memoria que se debe interpretar y además existen formas de direccionar esta memoria y ordenarla. Para acceder a memoria existen las siguientes formas, mediante:[2]

- Nivel de byte (8 bits)
- Nivel de media palabra (16 bits)

- Nivel de palabra / word (32 bits)
- Nivel de doble palabra (64 bits)

Además, la forma de organizar la memoria se da mediante un ordenamiento Little Endian o Big Endian. El primer ordenamiento, consisten en ordenar los bytes menos significativos en la dirección más pequeños de la palabra. Mientras que el segundo, ordena los bytes más significativos en la parte más pequeña de la palabra. Estos se pueden ejemplificar con la siguiente figura.

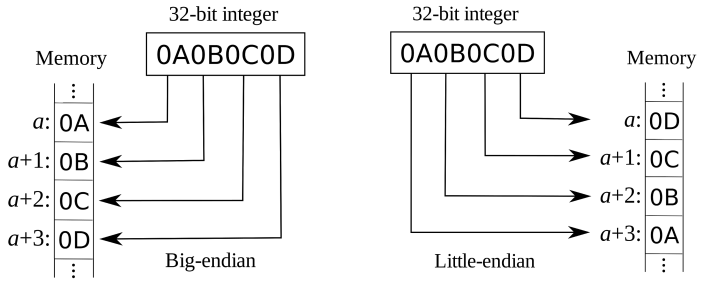

Figura 5. Big Endian y Little Endian

Bibliografía

[1] A. Sanchez Iglesias, "¿Qué es el conjunto de instrucciones de un procesador?", *about en español*, 2016. [Online]. Available: http://computadoras.about.com/od/conoce-procesadores/a/Conjunto-De-Instrucciones-Procesador.htm. [Accessed: 05- Sep- 2016].

[2] Camacho, R. (2012, 17 de Marzo). *Computo Integrado: Arquitectura RISC y CISC* Recuperado el 30 de Septiembre del 2016, de http://rcmcomputointegrado.blogspot.com/2012/03/arquitectura-risc-y-cisc.html

[3] D. Harris and S. Harris, *Digital design and computer architecture*. Amsterdam: Morgan Kaufmann Publishers, 2007.

[4] J. Hennessy, D. Patterson and A. Arpaci-Dusseau, *Computer architecture*. Amsterdam: Elsevier/Morgan Kaufmann Publishers, 2007.

[5] "MIPS Reference Card", 2016. [Online]. Available: https://inst.eecs.berkeley.edu/~cs61c/resources/MIPS_Green_Sheet.pdf. [Accessed: 05- Sep- 2016].

[6] P. Chen, Y. Chuang and V. Pietikainen, *ARM Architecture: Computer Organization and Assembly Languages*, 1st ed. 2016.

[7] Vega, J., Sánchez, R., Salgado, G. & Sánchez, L. (s. f.). *Arquitectura RISC vs CISC* Recuperado el 29 de septiembre del 2016, de http://www.azc.uam.mx/publicaciones/enlinea2/num1/1-2.htm

Anexos

Comparación de propuestas para implementación.

Diseñar un conjunto de instrucciones envuelve una gran cantidad de aspectos a considerar con el fin de generar un ISA optimizado que pueda ser ejecutado adecuadamente en el procesador.

Para este proyecto se efectuaron dos propuestas de un conjunto de instrucciones con el fin de compararlos para escoger el más óptimo para su uso posterior en el proyecto. Algunos aspectos del conjunto de instrucciones son iguales entre las propuestas. A continuación estos aspectos se detallan y justifican.

El primer atributo del conjunto de instrucciones es el nombre, para este ISA se escogió el nombre Coloso.

El segundo atributo a considerar es la clase del ISA, esto está relacionado con la forma en que almacenan y referencian los operandos con los que el procesador trabaja. Para este conjunto de instrucciones se seleccionó la clase Registro-Registro. Esto implica que todas las operaciones se realizan con registros en vez de memoria del sistema.

La razón por la cual se escogió el uso de la clase Registro-Registro, es porque de este modo se aumenta la velocidad de ejecución de instrucciones al no tener que accesar la memoria principal constantemente, la cual es de acceso más lento. Además la clase Registro-Registro es la clase por defecto de las arquitecturas de tipo RISC (Reduced Instruction Set Computing).[1]

El tercer aspecto a determinar es el tamaño de las instrucciones del conjunto de instrucciones. Puesto que esta arquitectura es de tipo RISC todas las instrucciones deben de ser del mismo tamaño. Para este ISA se definió un tamaño fijo de instrucciones de 32 bits, también conocido como de una palabra.

La razón por la cual se escogió un tamaño de instrucción de una palabra es para soportar el uso de números lo suficientemente grandes para uso en programas informáticos, pero sin ocupar demasiado espacio (cómo instrucciones de doble palabra) de *hardware.*

El cuarto aspecto a definir son los modos de direccionamiento, para este ISA el equipo acordó permitir el uso de cuatro modos de direccionamiento: Registro, Inmediato, Desplazamiento y Directo.

Esto se definió de esa manera para que se tenga una variada capacidad de escribir las instrucciones con los operandos. Además de esta forma se provee flexibilidad para realizar operaciones en donde se deben utilizar tanto registros como inmediatos.

El quinto aspecto de este ISA es definir la cantidad de registros, así como los registros de propósito general y los de propósito específico. Para este conjunto de instrucciones el equipo definió utilizar dieciséis registros, En donde trece son de propósito general y tres son de propósito específico.[4]

Los primeros trece registros que son de propósito general son utilizados para almacenar variables de los programas o para otro uso que un programador usando esta arquitectura quiera darle.

Los otros tres registros son los siguientes: El primer registro se utiliza para apuntar a memoria a la posición más actualizada del *stack* (pila de memoria) para uso en un programa. El segundo registro se utiliza para llevar el orden de las instrucciones a ejecutar, conocido como: program counter (contador de programa). El tercer registro es para almacenar una dirección de una función que llama a otra, eso es útil para poder crear funciones independientes, este registro es conocido como: link register (registro de vínculo).[1]

Las demás características y aspectos del ISA son diferentes entre las propuestas, a continuación se presentan ambas propuestas y sus diferencias, y se justifica la elección de uno de los dos.

Propuesta 1

La propuesta número uno define las siguientes instrucciones para cada categoría:

Aritméticas

- Add
- Sub
- Multiply
- Add unsigned
- Sub unsigned

Lógicas

- And
- Or
- Xor
- Nor

Control de flujo

- Branch on equal
- Branch greater than
- Branch greater equal than
- Jump
- Branch and link

Memoria

- Store register
- Load register
- Store byte
- Load byte

Desplazamiento

- Logical shift right
- Logical shift left
- Init

Este conjunto de instrucciones soporta los tipos de instrucciones más comunes de las arquitecturas tipo RISC. Las cuales son relativamente simples y permiten la creación de programas complejos.

Las operaciones aritméticas son: suma, resta, multiplicación, suma sin signo (operandos positiva), y resta sin signo (operandos positivos). Partiendo de estas operaciones se pueden componer expresiones matemáticas más complejas. Para las operaciones aritméticas se pueden utilizar los modos de direccionamiento: registro, inmediato, desplazamiento y directo.

Las operaciones lógicas son: AND lógico, OR lógico, XOR lógico y NOR lógico. Cada una de estas operaciones es a nivel bits, lo que significa que se toman

los operandos y se opera bit por bit. Partiendo de estas operaciones se pueden componer funciones lógicas más complejas. Para las operaciones lógicas se pueden utilizar los modos de direccionamiento: registro, inmediato, desplazamiento y directo.

Las operaciones de control de flujo son: ramificar cuando los operandos tienen el mismo valor, ramificar cuando un operando es mayor que el otro, ramificar cuando un operando es mayor que otro o sean del mismo valor, ramificar incondicionalmente y ramificar incondicionalmente almacenando la dirección de la instrucción. Con estas instrucciones se puede efectuar un control completo del flujo de un programa. Para las operaciones de control de flujo se pueden utilizar los modos de direccionamiento: registro, inmediato, desplazamiento y directo.

Las operaciones de control de memoria son: almacenar un registro en memoria, cargar datos de un registro de memoria a un registro, almacenar en memoria un byte y cargar a un registro un byte desde memoria. Con estas instrucciones el programador puede tener control sobre los datos cargados en registros y los datos almacenados en memoria. Para las operaciones de control de memoria se pueden utilizar los modos de direccionamiento: registro, inmediato, desplazamiento y directo.

Las operaciones de desplazamiento de datos son: corrimiento lógico de bits hacia la derecha, corrimiento lógico de bits hacia la izquierda e inicialización o carga de datos en un registro. Partiendo de estas operaciones se pueden realizar procedimientos básicos de desplazamientos de bits. Para las operaciones de desplazamiento se pueden utilizar los modos de direccionamiento: registro, inmediato, desplazamiento y directo.

Como se mencionó anteriormente la encodificación propuesta es de 32 bits, esto afecta a todas las instrucciones ya que la arquitectura es RISC. En esta propuesta se definió darle a cada instrucción un código de operación binario único, puesto que son veintiún instrucciones se necesitan cinco bits para darle a cada instrucción un código. En esta propuesta el código de operaciones es llamado Opcode.

Anteriormente también se definió que se van a utilizar cuatro modos de direccionamiento, entonces en esta encodificación se utilizan dos bits para indicar el tipo de direccionamiento, e.g. directo, inmediato, desplazamiento o directo. En esta propuesta los bits de encodificación son llamados Encode

En esta arquitectura se definió el uso de dieciséis registros, por tanto se necesitan cuatro bits para accesar a cada uno de ellos. Cada dirección de registro en este ISA tiene cuatro bits.

Finalmente se propusieron dos formatos de instrucciones. El primer formato está diseñado para las instrucciones con direccionamiento registro y el segundo formato está diseñado para las instrucciones con direccionamiento inmediato, desplazamiento y directo.

Formato #1 de instrucción

Opcode	Encode	Reg. Destino	Reg. Fuente 1	Reg. Fuente 2
(5 bits)	(2 bits)	(4 bits)	(4 bits)	(4 bits)

Formato #2 de instrucción

Opcode	Encode	Reg. Destino	Reg. Fuente 1	Inmediato
(5 bits)	(2 bits)	(4 bits)	(4 bits)	(17 bits)

Propuesta 2

La propuesta número dos define las siguientes instrucciones para cada categoría:

Aritméticas

- Add
- Sub
- Multiply
- Add unsigned
- Sub unsigned
- Multiply unsigned

Lógicas

- And
- Or
- Xor
- Nor

Control de flujo

- Branch on equal
- Branch greater than
- Branch greater equal than
- Jump
- Branch and link

Memoria

- Store register
- Load register
- Store byte
- Load byte
- Store half register
- Load half register

Desplazamiento

- Logical shift right
- Logical shift left
- Init

Este conjunto de instrucciones soporta los tipos de instrucciones más comunes de las arquitecturas tipo RISC. Las cuales son relativamente simples y permiten la creación de programas complejos.

Las operaciones aritméticas son: suma, resta, multiplicación, suma sin signo (operandos positiva), resta sin signo (operandos positivos) y multiplicación sin signo (operandos positivos). Partiendo de estas operaciones se pueden componer expresiones matemáticas más complejas. Para las operaciones aritméticas se pueden utilizar los modos de direccionamiento: registro, inmediato, desplazamiento y directo.

Las operaciones lógicas son: AND lógico, OR lógico, XOR lógico y NOR lógico. Cada una de estas operaciones es a nivel bits, lo que significa que se toman los operandos y se opera bit por bit. Partiendo de estas operaciones se pueden componer funciones lógicas más complejas. Para las operaciones lógicas se pueden utilizar los modos de direccionamiento: registro, inmediato, desplazamiento y directo.

Las operaciones de control de flujo son: ramificar cuando los operandos tienen el mismo valor, ramificar cuando un operando es mayor que el otro, ramificar

cuando un operando es mayor que otro o sean del mismo valor, ramificar incondicionalmente y ramificar incondicionalmente almacenando la dirección de la instrucción. Con estas instrucciones se puede efectuar un control completo del flujo de un programa. Para las operaciones de control de flujo se pueden utilizar los modos de direccionamiento: registro, inmediato, desplazamiento y directo.

Las operaciones de control de memoria son: almacenar un registro en memoria, cargar datos de un registro de memoria a un registro, almacenar en memoria un byte, cargar a un registro un byte desde memoria, almacenar medio registro en memoria y cargar medio registro de memoria a un registro. Con estas instrucciones el programador puede tener control sobre los datos cargados en registros y los datos almacenados en memoria. Para las operaciones de control de memoria se pueden utilizar los modos de direccionamiento: registro, inmediato, desplazamiento y directo.

Las operaciones de desplazamiento de datos son: corrimiento lógico de bits hacia la derecha, corrimiento lógico de bits hacia la izquierda e inicialización o carga de datos en un registro. Partiendo de estas operaciones se pueden realizar procedimientos básicos de desplazamientos de bits. Para las operaciones de desplazamiento se pueden utilizar los modos de direccionamiento: registro, inmediato, desplazamiento y directo.

Como se mencionó anteriormente la encodificación propuesta es de 32 bits, esto afecta a todas las instrucciones ya que la arquitectura es RISC. En esta propuesta se definió identificar el tipo de instrucción e incluirlo en la encodificación. Puesto hay cinco tipos instrucciones: aritméticas, lógicas, control de flujo, desplazamiento y control de memoria se necesitan tres bits para identificar a cada tipo de instrucción. En esta encodificación el tipo de instrucción se llama Tipo.

Para identificar a cada instrucción individualmente también se necesitan cuatro bits. Ya que con los bits de Tipo solamente se determina el tipo de instrucción. Los cuatro bits utilizados para identificar a cada instrucción son llamados Opcode.

Anteriormente también se definió que se van a utilizar cuatro modos de direccionamiento, entonces en esta codificación se utilizan dos bits para indicar el tipo de direccionamiento, e.g. directo, inmediato, desplazamiento o directo. En esta propuesta los bits de codificación son llamados Encode.

En esta arquitectura se definió el uso de dieciséis registros, por tanto se necesitan cuatro bits para acceder a cada uno de ellos. Cada dirección de registro en este ISA tiene cuatro bits.

Finalmente se propusieron dos formatos de instrucciones. El primer formato está diseñado para las instrucciones con direccionamiento registro y el segundo formato está diseñado para las instrucciones con direccionamiento inmediato, desplazamiento y directo.

Formato #1 de instrucción

Opcode	Tipo	Encode	Reg. Destino	Reg. Fuente 1	Reg. Fuente 2
(4 bits)	(3 bits)	(2 bits)	(4 bits)	(4 bits)	(4 bits)

Formato #2 de instrucción

Opcode	Tipo	Encode	Reg. Destino	Reg. Fuente 1	Inmediato
(4 bits)	(3 bits)	(2 bits)	(4 bits)	(4 bits)	(15 bits)

Comparación entre las dos propuestas

Después de comparar ambas opciones se optó continuar por la segunda propuesta por una serie de razones las cuales son detalladas a continuación.

- Es más fácil determinar el tipo de instrucción con la segunda propuesta. Esta ventaja estructural provee mayor simplicidad para mejoras futuras al conjunto de instrucciones.

- El conjunto de instrucciones de la segunda propuesta permite aumentar la capacidad de operaciones que esta arquitectura podría tener en el futuro. Ya que cada tipo de instrucción puede tener hasta dieciséis instrucciones.

- Al proveer una forma de discernir entre el tipo de instrucción puede simplificar el *hardware* de control del microprocesador que se va a implementar en las etapas posteriores del proyecto.

- La segunda propuesta además incluye más operaciones en el área de instrucciones de control de memoria, las cuales no están incluidas en la primera propuesta.

- La primer propuesta permite utilizar operadores inmediatos de diecisiete bits mientras que la segunda propuesta permite utilizar operadores de quince bits. A pesar de que el primer conjunto de instrucciones soporta números más grandes, no se consideró como una ventaja considerable ya que quince bits permite números relativamente grandes.

Coloso. Hoja de Referencia

Cd	Registro destino
Cs, Ct	Registros fuente
PC	Program Counter
Addr	Dirección de salto
OFF15	Offset de la posición
DIR15	Dirección de memoria

Operaciones Aritméticas	
ADD Cd, Cs, Ct	Cd = Cs + Ct (Overflow)
ADD Cd, Cs, !Imm_15	Cd = Cs + Imm15 (Overflow)
SUB Cd, Cs, Ct	Cd = Cs – Ct (Overflow)
SUB Cd, Cs, !Imm_15	Cd = Cs – Imm_15 (Overflow)
ADDUN Cd, Cs, Ct	Cd = Cs + Ct
ADDUN Cd, Cs, !Imm_15	Cd = Cs + Imm15
SUBUN Cd, Cs, Ct	Cd = Cs – Ct
SUBUN Cd, Cs, !Imm_15	Cd = Cs – Imm_15
MUL Cd, Cs, Ct	Cd = Cs * Ct
MUL Cd, Cs, !Imm_15	Cd = Cs * Imm15

Operaciones Lógicas	
AND Cd, Cs, Ct	Cd = Cs & Ct
AND Cd, Cs, !Imm_15	Cd = Cs & Imm_15
OR Cd, Cs, Ct	Cd = Cs | Ct
OR Cd, Cs, !Imm_15	Cd = Cs | Imm_15
NOR Cd, Cs, Ct	Cd = ~ (Cs | Ct)
NOR Cd, Cs, !Imm_15	Cd = ~ (Cs & Imm_15)
XOR Cd, Cs, Ct	
XOR Cd, Cs, !Imm_15	

Operaciones Control de Flujo	
BREQ Cs, Ct, Addr	Si Cs == Ct PC += Addr
BRGT Cs, Ct, Addr	Si Cs > Ct PC += Addr
BRGTEQ Cs, Ct, Addr	Si Cs >= Ct PC += Addr
JUMP Addr	PC = $PC_{31:28}$::Addr
BRLN Addr	C15 = PC + 8; PC = Addr

Operaciones Memoria	
STOREREG Ct, (OFF15, Cs)	MEM32[Cs + OFF15] = Ct
STOREREG Ct, (DIR15)	MEM32[DIR15] = Ct
LOADREG Ct, (OFF15, Cs)	Ct = MEM32[Cs + OFF15]
LOADREG Ct, (DIR15)	Ct = MEM32[DIR15]
STOREBYTE Ct, (OFF15, Cs)	MEM16[Cs + OFF15] = $Ct_{7:0}$
STOREBYTE Ct, (DIR15)	MEM16[DIR15] = $Ct_{7:0}$
LOADBYTE Ct, (OFF15, Cs)	Ct = MEM8[Cs + OFF15]
LOADBYTE Ct, (DIR15)	Ct = MEM8[DIR15]
STOREHALF Ct, (OFF15, Cs)	MEM16[Cs + OFF15] = $Ct_{15:0}$
STOREHALF Ct, (DIR15)	MEM16[DIR15] = $Ct_{15:0}$
LOADHALF Ct, (OFF15, Cs)	Ct = MEM16[Cs + OFF15]
LOADHALF Ct, (DIR15)	Ct = MEM16[OFF15]

Operaciones Desplazamiento	
SHIFTR Cd, Shamt (Cs)	Cd = Cs >> Shamt
SHIFTL Cd, Shamt (Cs)	Cd = Cs << Shamt
INIT Cs, Ct	Cs = Ct

Mnemónico	**Significado**
Add	Suma con signo
Sub	Resta con signo
Multiply	Multiplicación
Add unsigned	Suma sin signo
Sub unsigned	Resta sin signo
Multiply unsigned	Multiplicación sin signo
And	Operación Lógica Y
Or	Operación Lógica O
Xor	Operación Lógica O excluyente
Nor	Operación Lógica O Negada
Branch on equal	Salto condicional Igual
Branch greater than	Salto condicional Mayor que
Branch greater equal than	Salto condicional Mayor Igual que
Jump	Salto Incondicional
Branch and link	Salto y enlace
Load register	Cargar registro
Store register	Almacenar registro
Store byte	Almacenar Registro
Load byte	Cargar Byte
Store half register	Almacenar Media Palabra
Load half register	Cargar Media Palabra
Logical shift right	Desplazamiento a la derecha
Logical shift left	Desplazamiento a la izquierda
Init	Intercambio