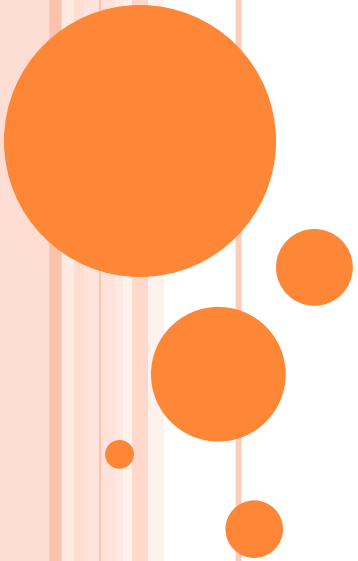
ISA



Instruction Set Architecture

Agenda

- Definiciones
- ARC
- Memoria
- Endianness
- Camino de datos (Data Path)
- Formato de Instrucción
- Hola Assembler
- Direcciones de 32bits
- Subrutinas

Definiciones

- ARC significa **A** **R**ISC **C**omputer
- RISC significa **R**educed **I**nstruction **S**et **C**omputer
- CISC significa **C**omplex **I**nstruction **S**et **C**omputer
- ISA significa **I**nstruction **S**et **A**rchitecture

Definiciones

CISC

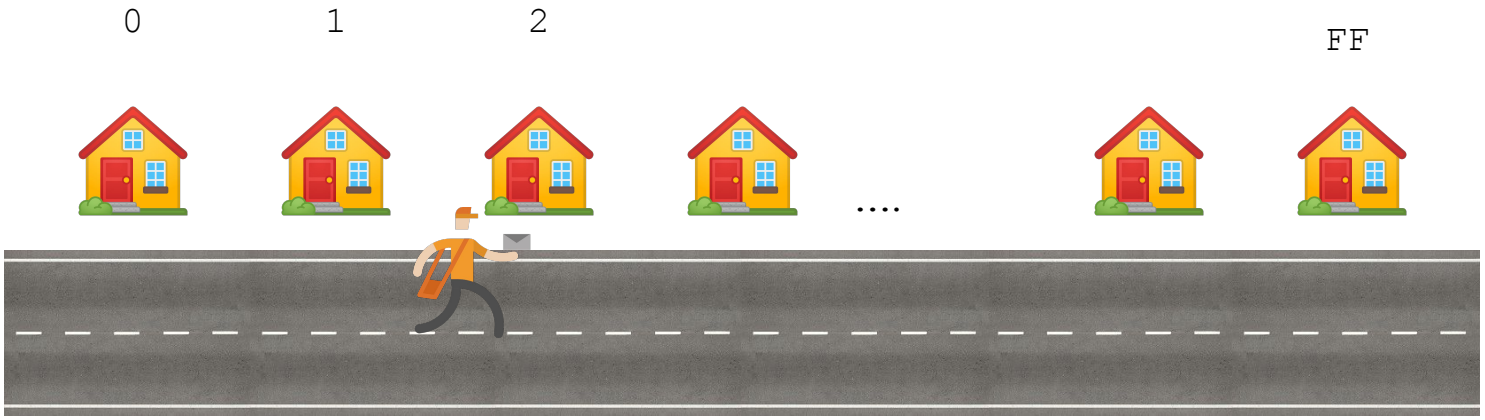
RISC

- Computadoras hogareñas
- Instrucciones muy amplias
- Cualquier instrucción puede acceder a memoria
- Ventajas
  - Crear compiladores
  - Depuración de errores

- Entornos de red
- Instrucciones de tamaño fijo
- Acceso a memoria solo a través de Load y Store
- Ventajas
  - Segmentación y paralelismo
  - Menos de ciclos de clock

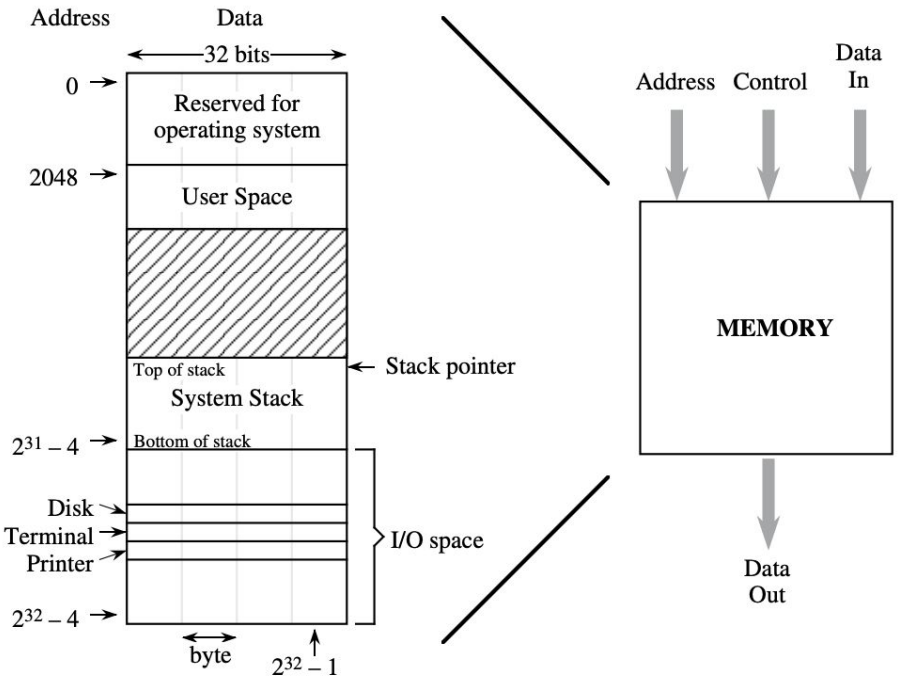
Memoria

Una memoria es un arreglo lineal de distintas locaciones de memorias ordenadas en forma consecutiva



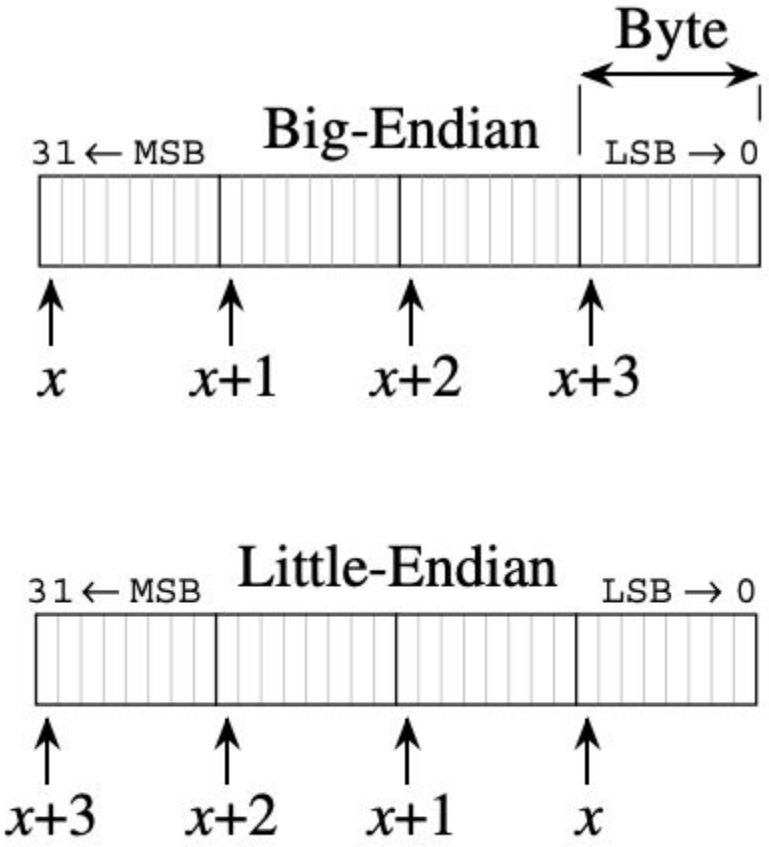
- Dirección: Es el número que identifica cada “casa” (locación de memoria). ¿A donde debe llevar el paquete?
- Contenido: Es lo que está dentro de la “casa” (contenido de la memoria)
- Capacidad de almacenamiento: El tamaño de la “casa”. ¿Cuantos bits puedo guardar en esa posición de memoria?

Memoria en ARC



- Espacio de direcciones: 2^{32}
- Dato: 32 bits (4 bytes)
- Capacidad de direccionamiento: 1 Byte

Endianess


Byte
Big-Endian
31 ← MSB
LSB → 0
x
x+1
x+2
x+3
Little-Endian
31 ← MSB
LSB → 0
x+3
x+2
x+1
x

Ejercicio

1.- Una línea de transmisión de datos comunica la computadora “A”, que es “little endian”, con la computadora “B”, que es “big endian”. A través de esta línea se transmite desde la computadora A una palabra de 32 bits que contiene el valor numérico 3. Esto se hace byte por byte de modo que en la computadora B el byte 0 se almacena en el byte 0, el byte 1 en el byte1, etc.
Una vez terminada la transmisión ¿Cuál es el valor numérico leído en la máquina B?

Ejercicio

1.- Una línea de transmisión de datos comunica la computadora "A", que es "little endian", con la computadora "B", que es "big endian". A través de esta línea se transmite desde la computadora A una palabra de 32 bits que contiene el valor numérico 3. Esto se hace byte por byte de modo que en la computadora B el byte 0 se almacena en el byte 0, el byte 1 en el byte1, etc.
Una vez terminada la transmisión ¿Cuál es el valor numérico leído en la máquina B?

Valor: `00000003h`

A es little Endian

03h	0
00h	1
00h	2
00h	3

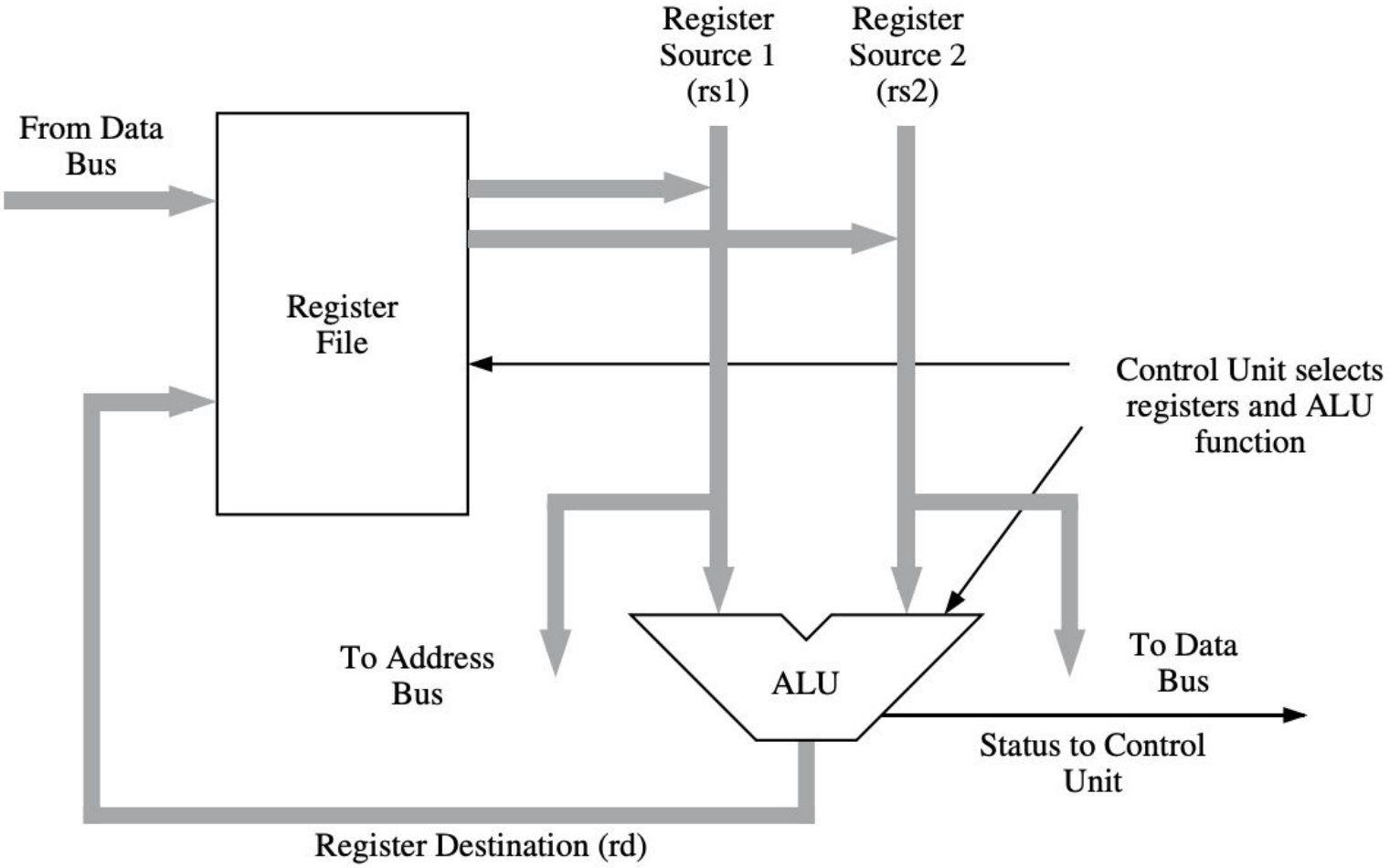
Valor leído: `00000003h`

B es Big Endian

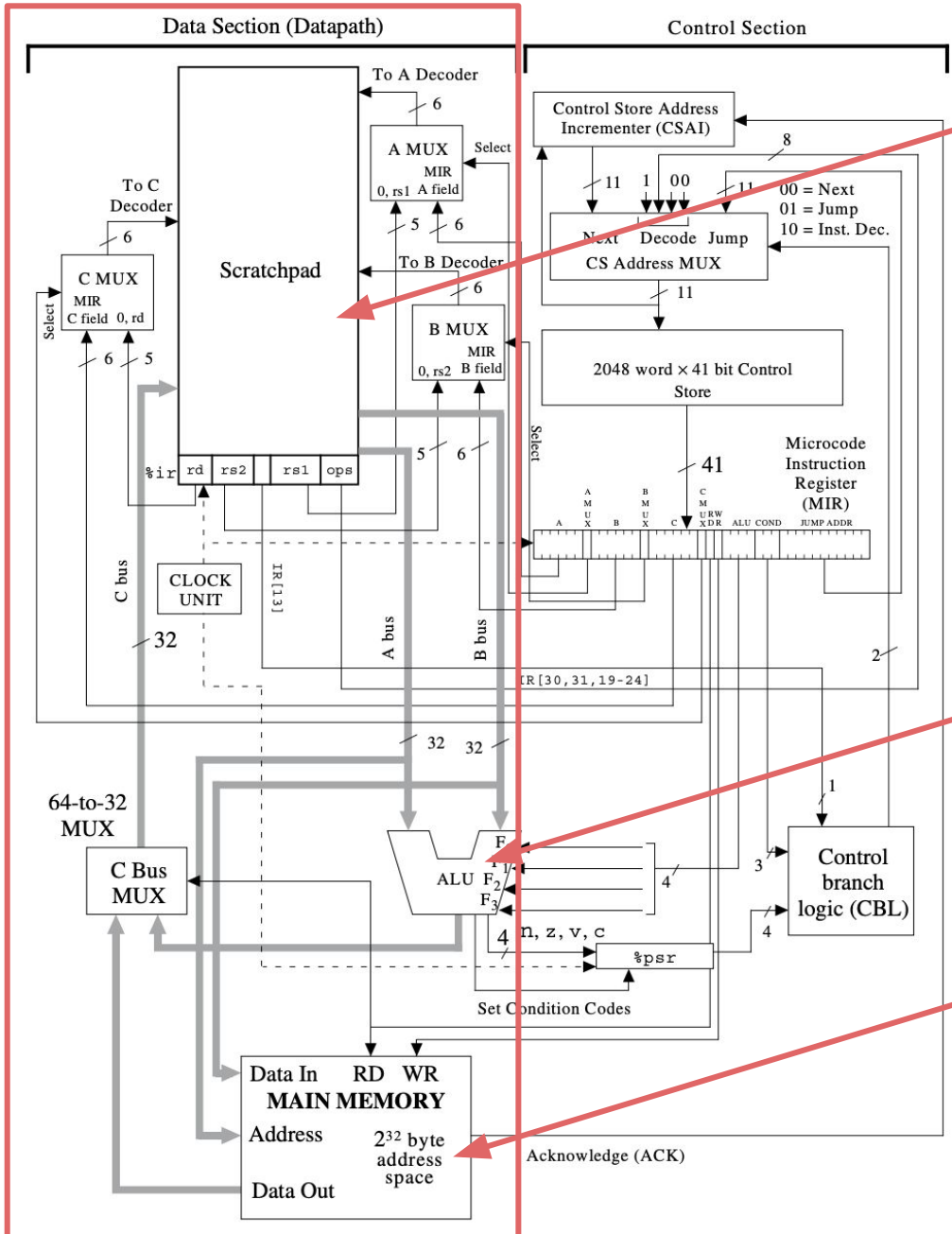
03h	0
00h	1
00h	2
00h	3

Valor leído: `03000000h`

Camino de Datos



Camino de Datos

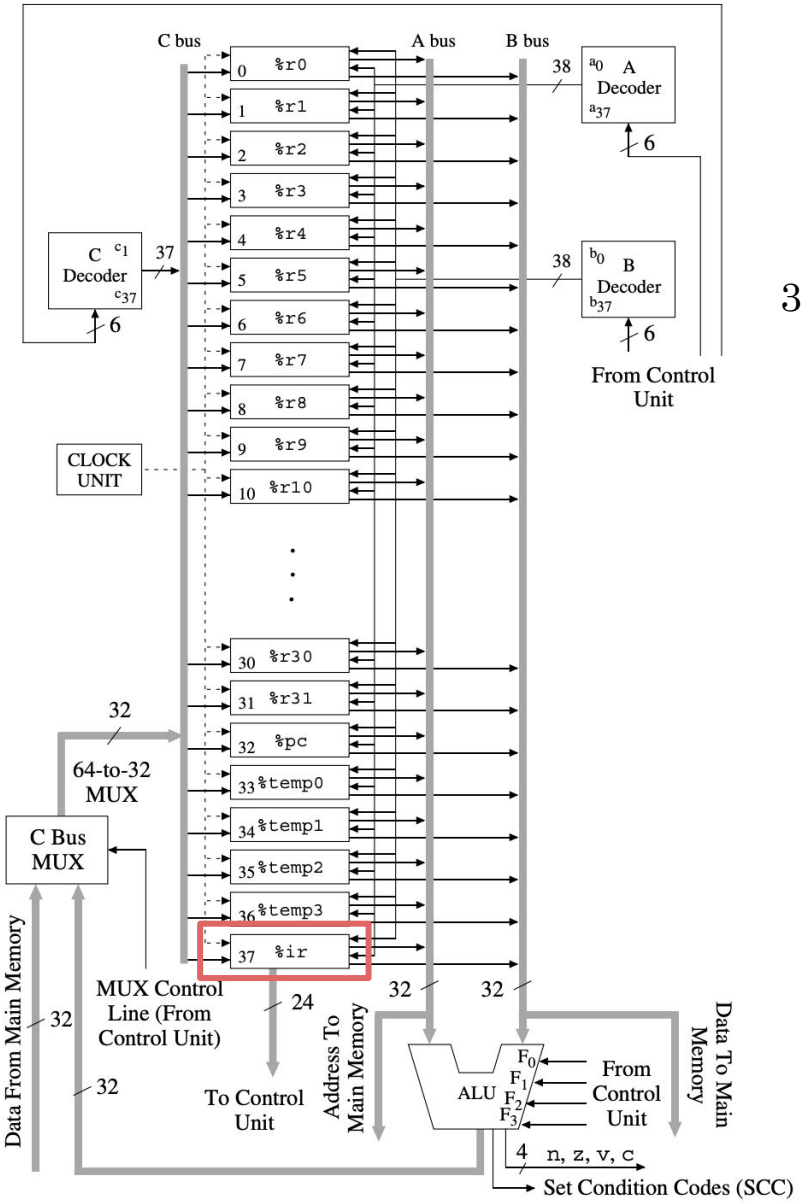


Register File o Archivo de registros

ALU

Memoria Principal

Register File



37 registros

32 registros de propósito general

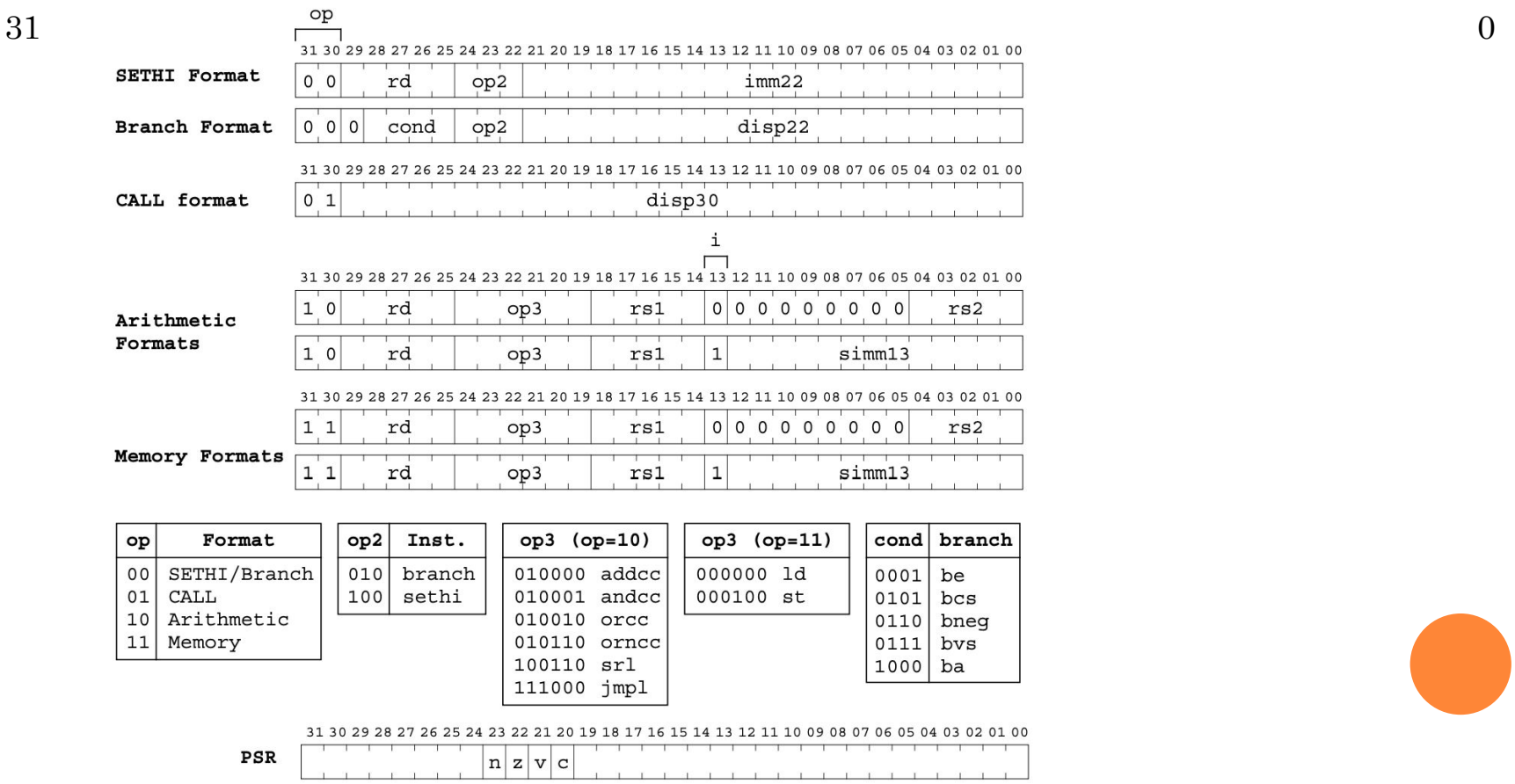
- del %r0 al %r31
- %r0 es siempre cero
- %r14 stack pointer o puntero de pila
- %r15 link register

5 registros de uso interno

- del %r32 al %r37
- %r32 program counter
- %r33-36 registros temporales
- %r37 registro de instrucción

Formato de Instrucción

%r37 Registro de instrucción



Ejercicio

30.- Un programa fue bajado de disco a memoria principal de una computadora ARC. En la tabla siguiente se muestra el contenido de un segmento de memoria dentro del rango ocupado por ese programa.

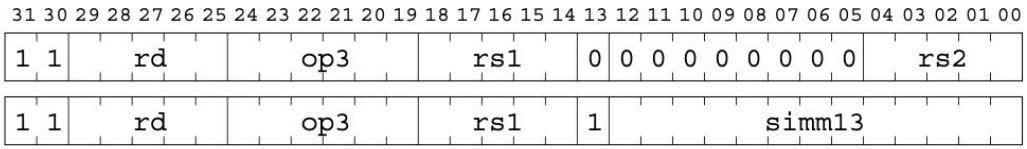
Dirección	Contenido
00000810	CA006BCC
00000814	CA206BB8
00000818	82206004
0000081C	02800003
00000820	80A06000
00000824	10BFFFFB

Indicar el código Assembler de ese segmento.

Ejercicio

- es el origen en un store
- es el destino en un load

Dirección	Contenido
00000810	CA006BCC



op3 (op=11)
000000 ld
000100 st

Instrucción

```
ld    %r1  +3020   ,%r5
```

Carga en %r5 el contenido de la dirección de memoria r1 + 3020 en el registro r5

Ejercicio

3.- El set de instrucciones ARC incluye la instrucción “ld” para leer de memoria RAM datos de 32 bits pero también ofrece instrucciones que permiten acceder a memoria para leer datos de 16 bits y de 8 bits. Notar que si bien existe una única instrucción para leer datos de 32 bits en los otros dos casos hay dos instrucciones: una para números con signo y otra números sin signo ("ldsh" y "lduh" para datos de 16 bits con y sin signo respectivamente; "ldsb" y "ldub" para datos de 8 bits con y sin signo respectivamente). Se pide:

(a) Justificar la existencia de instrucciones diferenciadas para leer números con y sin signo.

(b) En el caso de escritura de datos en RAM no se hace esta diferenciación entre números signados y no-signados (las instrucciones previstas para escritura en memoria son “st” “sth” y “stb” para escribir 32, 16 y 8 bits respectivamente). Justificar esta diferencia con respecto a lo observado en la operación de lectura.

Hola Assembler

	Mnemonic	Meaning
Memory	**ld**	Load a register from memory
	st	Store a register into memory
Logic	**sethi**	Load the 22 most significant bits of a register
	andcc	Bitwise logical AND
	orcc	Bitwise logical OR
	orncc	Bitwise logical NOR
	srl	Shift right (logical)
Arithmetic	**addcc**	Add
Control	**call**	Call subroutine
	jmpl	Jump and link (return from subroutine call)
	be	Branch if equal
	bneg	Branch if negative
	bcs	Branch on carry
	bvs	Branch on overflow
	ba	Branch always

Hola Assembler

Pseudo-Op	Usage	Meaning
.equ X	.equ #10	Treat symbol X as $(10)_{16}$
.begin	.begin	Start assembling
.end	.end	Stop assembling
.org	.org 2048	Change location counter to 2048
.dwb	.dwb 25	Reserve a block of 25 words
.global	.global Y	Y is used in another module
.extern	.extern Z	Z is defined in another module
.macro	.macro M a, b, ...	Define macro M with formal parameters a, b, ...
.endmacro	.endmacro	End of macro definition
.if	.if <cond>	Assemble if <cond> is true
.endif	.endif	End of .if construct

Hola Assembler

Estructura de un programa de assembler

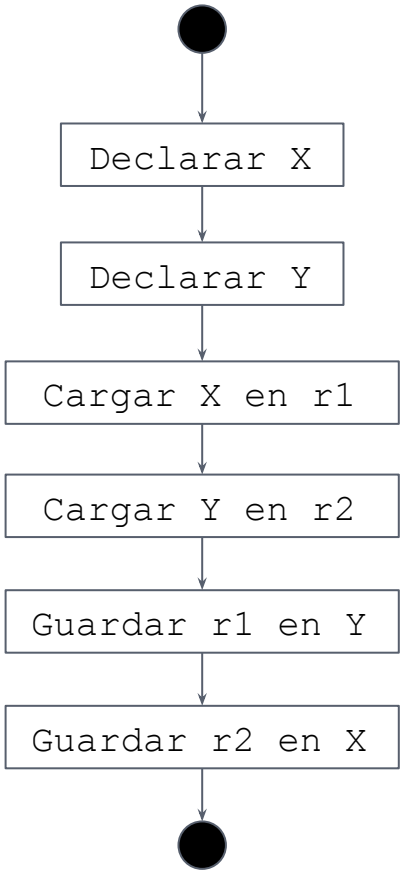
```
.begin
.org 2048
    !código assembler



.end
```

Ejercicio

4.- Escriba un programa en código ARC que declare dos variables de 32 bits en memoria RAM y que intercambie el contenido entre ellas. Utilizar el mínimo número de registros que le sea posible.

Ejercicio

4.- Escriba un programa en código ARC que declare dos variables de 32 bits en memoria RAM y que intercambie el contenido entre ellas. Utilizar el mínimo número de registros que le sea posible.




```
.begin
.org 2048

     ld X,%r0,%r1
     ld Y,%r0,%r2
     st %r1,%r0,Y
     st %r2,%r0,X

     X:25
     Y:45
.end
```

Direcciones de 32bits

Si las instrucciones son de 32bits, ¿Cómo cargamos una dirección de 32bits en un registro?

Usando la instrucción sethi

"`sethi`" establece los 22 bits más representativos y coloca ceros en los 10 bits menos significativos.

Ejemplo: 0xFF451200

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	0	1	0	0	0	1	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0

Los 22 bits menos significativos: 0x051200

Los siguientes 22bits serían:0x3FD144

Direcciones de 32bits

```
.begin
.org 2048
    !Posición 0xFF451200
    X22 .equ 0x051200
    X10 .equ 0x3FD144

    sethi X22,%r1
    sethi X10,%r2
    srl %r1,10,%r1
    or %r1,%r2,%r3

.end
```

Direcciones de 32bits

Declarando la dirección en memoria, dentro inferior al rango 4096

```
.begin
.org 2048
    !Posición 0xFF451200
    X: 0xFF451200

    ld X,%r0,%r1
.end
```

Direcciones de 32bits

Utilizando la instrucción call

```
.begin
.org 2048

    call 0xFF451200

seguir: add %r1,0,%r1

.org 0xFF451200

jmpl %r15,4,%r1

.end
```

Subrutinas

Se utilizan las instrucciones `call` y `jmpl`

Pasaje de parámetros

- Por registro
  - Los parámetros se guardan en los registros
  - La subrutina espera que los parámetros estén en dichos registros (ABI)
- Por stack
  - Los parámetros se pushean a la pila
  - La subrutina hace un pop de la pila para obtener los parámetros
- Por área reservada en memoria
  - Se reserva un espacio en memoria. En dicho espacio se guardan los parámetros. Se pasa por registro el puntero a la posición del área de memoria.
  - La subrutina accede al área de memoria para obtener los parámetros

Ejercicio

6.- Un programa ARC declara dos variables y luego invoca una rutina que obtiene la suma de ambas. Escribir tres versiones del programa principal y de la rutina considerando diferentes convenciones para el pasaje de parámetros (a) por registros (b) por pila (c) por área reservada de memoria
La subrutina debe ser declarada en el mismo módulo que el programa principal.

¿Preguntas?