Temas:

Modelo de Bus de Sistema. Computadora ARC. La arquitectura de programación. Los lenguajes y la máquina.

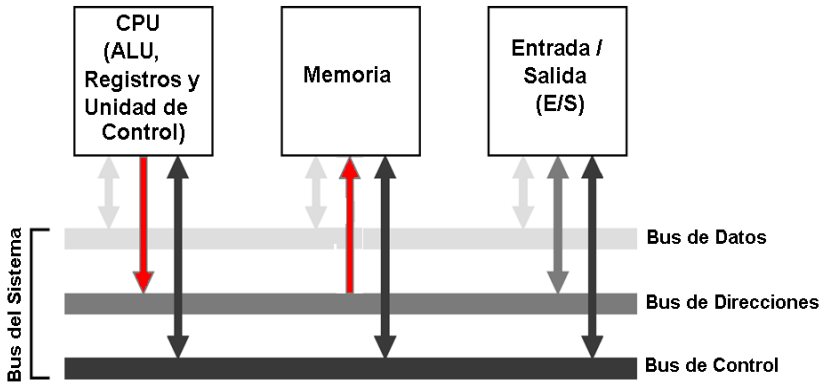
OBJETIVOS:

- Conocer los bloques básicos de una computadora, mediante el Modelo de Bus de Sistemas.
- Explicar el ciclo de búsqueda y ejecución.
- Presentar la arquitectura de programación ARC.
- Estudiar algunas propiedades generales de la arquitectura de programación.
- Establecer la forma en que se calcula la dirección cuando se accede a memoria.
- Estudiar algunas instrucciones.
- Analizar la forma en que estas secuencias de instrucciones se traducen en código objeto.

BIBLIOGRAFÍA:

- Murdocca y Heuring, “Principios de Arquitectura de Computadoras”, Prentice Hall, (capítulo 1, 4 y 5).
- Material del aula virtual de la cátedra.
- Otros.

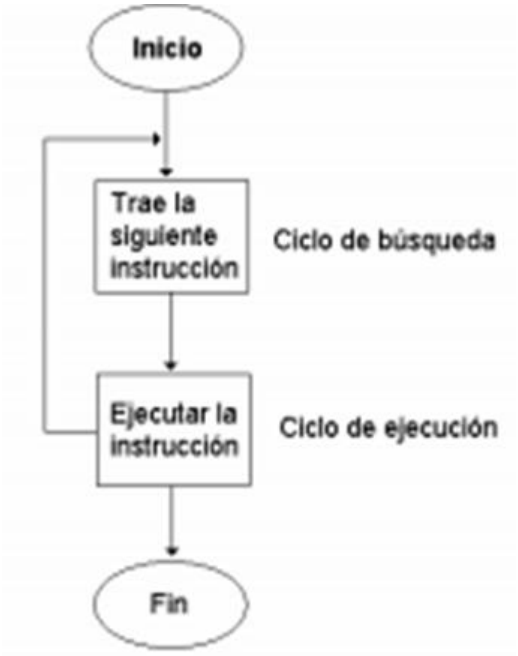
1 – EL MODELO DEL BUS DEL SISTEMA



La CPU genera direcciones que se transfieren sobre el bus de direcciones.
La Memoria recibe esas direcciones a través del mismo bus.
La Memoria nunca genera direcciones y la CPU nunca recibe direcciones.

2- CICLO DE INSTRUCCIÓN

Vista simplificada de un ciclo de instrucción:



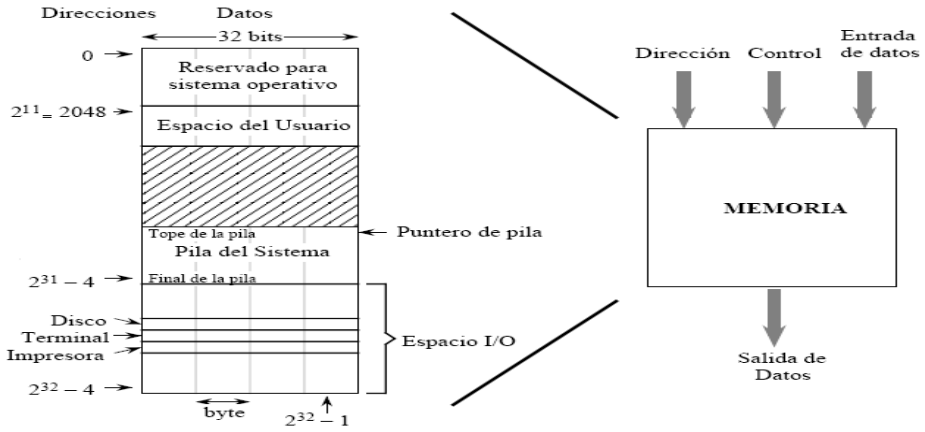
3 - NIVELES DE MÁQUINA EN LA JERARQUÍA DE UN SISTEMA

Superior	Nivel de usuario: programas de aplicación.
	Lenguajes de alto nivel.
	Lenguajes de bajo nivel (assembler y código máquina)
	Control microprogramado / cableado
	Unidades funcionales (ALU, memoria, etc.)
	Circuitos lógicos
Inferior	Transistores y cables

4- LA ARQUITECTURA ARC.

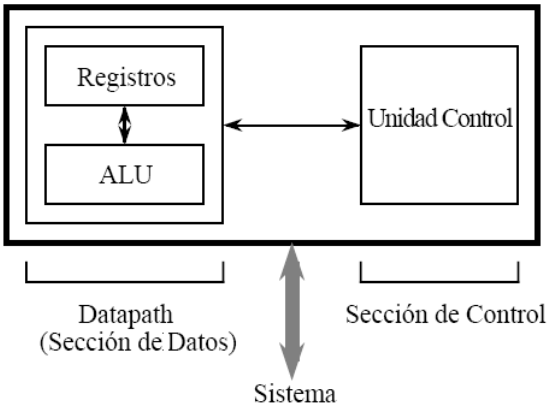
4.1 – La memoria en ARC.

ARC (A RISC Computer) es un subconjunto del modelo de arquitectura basado en el procesador SPARC, desarrollado por Sun Microsystems.
En una máquina direccionable por byte, el dato más pequeño que se puede buscar en memoria es el byte. Las palabras multi bytes se almacenan como secuencias de bytes, y se direccionan a partir del byte menos significativo de la palabra almacenada.
Las direcciones de memoria están ordenadas en forma consecutiva. Cada locación numerada corresponde a una palabra en ARC. El único número que identifica a cada palabra se conoce como su dirección.



4.2 - La CPU en ARC.

La CPU, Unidad Central de Procesamiento, consiste de una sección de datos que contiene registros y una ALU, y una sección de control, que interpreta las instrucciones y realiza las transferencias entre registros. La sección de datos se conoce como "camino de datos" o "datapath".
La CPU lee las instrucciones y los datos desde la memoria, ejecuta las instrucciones y almacena los resultados nuevamente en la memoria.



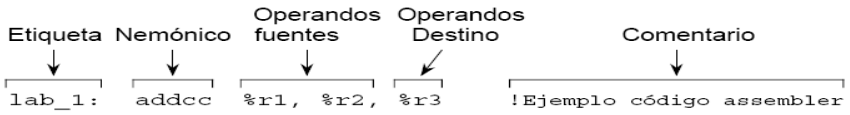
4.3 – El conjunto de instrucciones en ARC.

Es la colección de instrucciones que un procesador puede ejecutar.
Difiere de un procesador a otro, en el tamaño de las instrucciones, el tipo de operaciones que permiten, el tipo de operandos que puede ejecutar y los resultados que pueden entregar.
El tamaño de una instrucción en ARC es de 32 bits, o sea una palabra.

	Nemonico	Significado
Memoria	ld	Cargar registro desde la memoria
	st	Almacenar un registro en la memoria
Lógicas	sethi	Cargar los 22 bits mas significativos de un registro
	andcc	Operación lógica AND bit a Bit
	orcc	Operación lógica OR bit a bit
	orncc	Operación lógica NOR bit a bit
	srl	Desplazar a derecha (lógico)agrega ceros a la izq.
Aritmeticas	addcc	Sumar
Control	call	Salto ó Llamado a subrutina
	jmpl	Salto y Enlace (retorno de subrutina)
	be	Bifurcación o Salto por igual
	bneg	Bifurcación o Salto por negativo
	bcs	Bifurcación o Salto por acarreo
	bvs	Bifurcación o Salto por desborde u "overflow"
	ba	Bifurcación o Salto incondicional

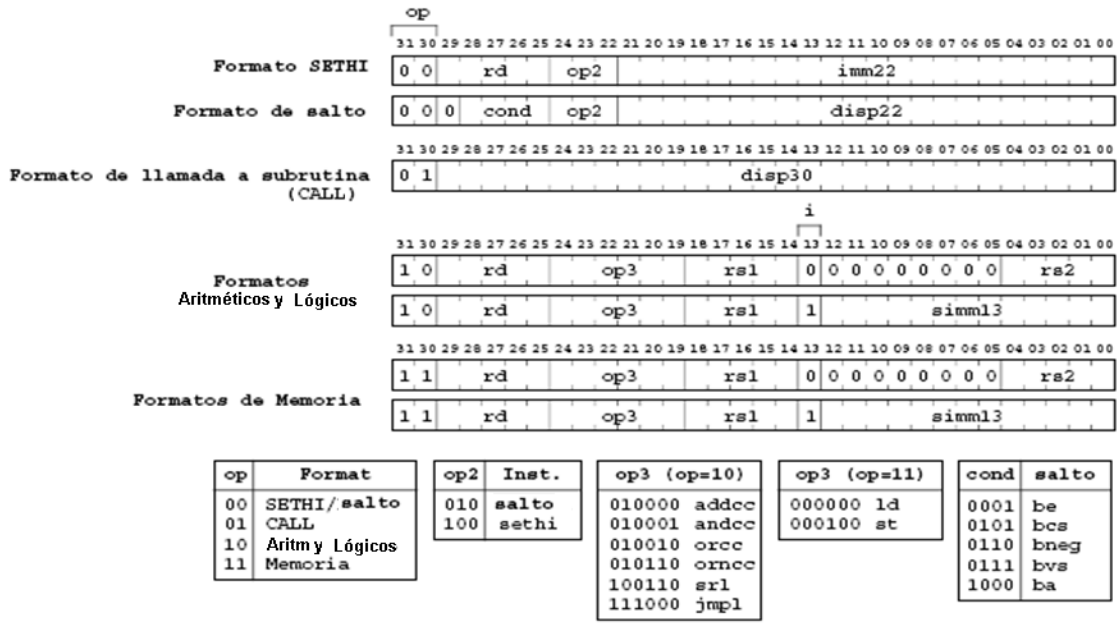
4.4– El formato del lenguaje ensamblador ARC.

El lenguaje hace distinción entre mayúsculas y minúsculas.
Los campos de etiqueta y comentario son optativos.
El campo etiqueta usa caracteres alfabéticos, numéricos (siempre y cuando no sea el primer dígito), los símbolos guión bajo (_), signo monetario ($), punto (.) y los dos puntos (:) que indica el final de la etiqueta.
El campo comentario va precedido del símbolo **!**.
Los operandos se separan con comas (,) y su uso dependerá de cada instrucción.



4.5– Formato de instrucciones en ARC.

El formato de instrucción definirá como el programa ensamblador (que traduce programas en lenguaje assembler a códigos binarios), distribuye los diferentes campos de una instrucción y la forma en que los interpreta la Unidad de Control.
Cada instrucción tiene 32 bits.
Los dos bits más significativos forman el campo OP, que corresponde al código de operación, a partir de éste se identificará el formato.



op	Format
00	SETHI/salto
01	CALL
10	Aritm y Lógicos
11	Memoria

op2	Inst.
010	salto
100	sethi

op3 (op=10)	
010000	addcc
010001	andcc
010010	orcc
010110	orncc
100110	srl
111000	jmpl

op3 (op=11)	
000000	ld
000100	st

cond	salto
0001	be
0101	bcs
0110	bneg
0111	bvs
1000	ba

4.6– El proceso de ensamblado.

Un compilador traduce un lenguaje de alto nivel, que es independiente de la arquitectura, a lenguaje ensamblador, el cual es dependiente de la arquitectura.
El proceso de transformar un programa en lenguaje ensamblador en un programa en lenguaje de máquina se conoce como proceso de ensamblado.
La mayoría de los ensambladores recorren dos veces el texto escrito en lenguaje simbólico, llamado **proceso de ensamblado en dos pasadas**.
En la primera pasada determinan:

- direcciones de todos los datos e instrucciones del programa.
- seleccionan que instrucción del lenguaje máquina debe generarse para cada instrucción en lenguaje simbólico, pero sin generar aún el código máquina.
- realizan operaciones aritméticas.
- insertan las definiciones de etiquetas y constantes en una tabla de símbolos. A cada símbolo (etiqueta) se le ingresa en la tabla el valor correspondiente a la posición en que se encuentran.

En la segunda pasada se genera el código máquina, insertando en el mismo los valores de los símbolos ya conocidos de la tabla anterior.

PROBLEMAS PROPUESTOS

1. Dibuje el modelo de Bus de Sistema. Explique las funciones de sus componentes.
2. En el Modelo de Bus de Sistema:
   A. ¿qué componente ejecuta las instrucciones?
   B. ¿cuál almacena los datos y e instrucciones listas para ejecutarse?
3. Explique cómo se ejecutarían los ciclos de búsqueda y ejecución en el modelo de Bus del Sistema.
4. Indique las características de las arquitecturas RISC.

5. Describa la memoria del computador ARC (tamaño, rango de direcciones, mapa de memoria, tipos y tamaños de las palabras).
6. Una palabra en ARC ¿cuántos registros usa?, ¿de cuántos bytes cada uno?

7. La CPU en ARC se la estudia separada en dos secciones:
   A. Bloque trayecto de datos: ¿Cómo está formado? ¿De qué otra forma se lo denomina? Dibújelo, indicando las funciones de los buses y demás elementos involucrados. ¿Cuántos registros de propósito general tiene? Particularice los registros 0, 14 y 15.
   B. Bloque de control: ¿Qué función cumple?

8. ¿Cuántos bits se usan para indicar la dirección de un registro de la CPU? ¿Por qué?
9. ¿Para qué sirven el contador de programa y el registro de instrucción?
10. ¿Cómo es el Ciclo de Búsqueda – Ejecución en ARC?, ¿para qué sirve? En el paso 1 del mismo ¿qué registros del CPU están involucrados? ¿En cuánto se debe incrementa el PC y por qué?
11. Indique diferencias entre registros del CPU y registros de la memoria RAM.

12. Explique cada subconjunto de instrucciones ARC.

13. ¿Qué es PSR? ¿Dónde se encuentra? ¿Qué instrucciones se relacionan con él? ¿Quiénes pueden modificarlo? Explique las funciones de los códigos de condición. ¿Cómo indicaría que el resultado es positivo?

14. ¿Cómo es el formato del lenguaje simbólico ARC?
15. ¿Qué son los formatos de instrucción?, ¿en qué se vinculan con el registro IR? En los formatos aritmeticológicos y de memoria ¿qué nos indica el bit 13?

16. Dados los siguientes enunciados, escriba las instrucciones en assembler y luego conviértalas a código máquina. Ejemplifíquelas dibujando mapa de memoria y registros de CPU involucrados, usted decide los valores no indicados.
   A. Leer el contenido de la dirección 2048 de memoria, use el registro 24 como destino.
   B. Escribir en la dirección 4096 de memoria el contenido del registro 24.
   C. Leer el contenido de la dirección 10000 de memoria, use el registro 22 como destino.
   D. Suma aritmética usando dos registros fuentes, y el registro 5 como destino.

E. Suma aritmética usando un registro fuente y una constante, y el registro 10 como destino.
F. Suma lógica usando dos registros fuentes y el registro 25 como destino. Los valores de los registros fuentes son $00000000_{(16)}$ y $FFFFFFFF_{(16)}$. Indique que valor queda almacenado en el registro destino.
G. Salto a subrutina, usando un valor, no etiqueta, para la cantidad de palabras a saltar.
H. Salto incondicional, ba 12, almacenada en la dirección 4000 de memoria. ¿Cuál sería la dirección hacia dónde salta?
I. Salto condicional. Indique bajo que condición saltaría.
J. Instrucción jmpl %r15 +4, %r0, almacenada en la dirección 4000. Si tomando como referencia que la última instrucción call esta almacenada en la dirección 2000, ¿cuál sería la nueva dirección del PC?.

17. ¿Qué son pseudo operaciones o directivas? Describa brevemente.
18. ¿Qué sucede cuando una sentencia, en un proceso de traducción, aparece antes del .begin o después del .end, para el lenguaje ensamblador de ARC?.
19. Un programa que está en lenguaje de máquina debe ser pasado a lenguaje simbólico. Una vez traducido, ¿con cuántas directivas, etiquetas y comentarios finales queda el programa traducido?
20. ¿Cuántas instrucciones a la vez ejecuta la CPU en ARC?, ¿en qué lenguaje deben estar esas instrucciones?
21. ¿Qué es un ensamblado de dos pasadas?

22. Crear una tabla de símbolos para el segmento de programa ARC que se muestra a continuación.

```
    ! Programa principal
    .begin
    .org        2048
    .extern  sub
main:  ld       [x],     %r2
       ld       [y],     %r3
       call     sub
       jmpl     %r15 + 4,     %r0
x:     105
y:     92
    .end
```

```
     ! Biblioteca de subrutina
    .begin
    ONE .equ   1
    .org 3000
    .global sub
sub:    orncc   %r3, %r0, %r3
        addcc %r3, ONE, %r3
        jmpl    %r15 + 4, %r0
     .end
```

23. Escribir una subrutina para ARC que realice una operación de intercambio entre los operandos de bits x = 25 e y = 50, los que se encuentran almacenados en memoria. Usar la menor cantidad de registros.
24. Desarrolle un programa en assembler que realice el cálculo: z = x + y. Pruebe el mismo en el Simulador ARC, que se encuentra en el aula virtual, copie y pegue las pantallas que surgen.
25. A continuación, se muestra una sección de código simbólico de ARC, responda:
    A. ¿Qué función cumple el programa? ¿Cómo lo hace? Realice una prueba de escritorio.
    B. Indique cuales son las etiquetas, las directivas, las instrucciones y los comentarios.
    C. Realice la tabla de símbolos.

D. Páselo a lenguaje máquina.

```
! Uso de los Registros:  %r1 - Longitud del arreglo a
!                        %r2 - Direcciónde inicio del arreglo a
!                        %r3 - LaSumaparcial
!                        %r4 - Puntero dentro del arreglo a
!                        %r5 - Contiene un elemento de a
            .begin                ! Comienzo del ensamblado
            .org  2048            ! Inicio del programa en 2048
a_start     .equ  3000            ! Dirección del arreglo a
            ld    [length], %r1   ! %r1 ← long. del arreglo a
            ld    [address],%r2   ! %r2 ← dirección de a
            andcc %r3, %r0, %r3   ! %r3 ← 0
loop:       andcc %r1, %r1, %r0   ! Verifica nº de elementos restantes.
            be    done            ! Finalizar cuando length=0
            addcc %r1, -4, %r1    ! Decrementar longitud arreglo
            addcc %r1, %r2, %r4   ! Dirección próximo elemento
            ld    %r4, %r5        ! %r5 ← Memoria[%r4]
            addcc %r3, %r5, %r3   ! Sumar nuevo elemento en r3
            ba    loop            ! Repetir lazo.

done:       jmpl  %r15 + 4, %r0   ! Retorno a rutina de llamada

length:       20                  ! 5 numeros (20 bytes) en a
address:      a_start
            .org  a_start         ! Inicio del arreglo a
a:              25                ! length/4 valores siguientes
               -10
                33
                -5
                 7
            .end                        ! Fin ensamblado
```

26. Un desensamblador es un programa que lee un módulo objeto y recrea el módulo fuente en lenguaje simbólico. Dado el siguiente código objeto, desensamblarlo para obtener las sentencias correspondientes del lenguaje simbólico ARC. Dado que el código objeto no contiene información suficiente para determinar los nombres de los símbolos, se les asignarán ordenadamente las letras del abecedario a medida que sean necesarias.

1100	0010	0000	0000	0010	1000	0001	0100
1100	0100	0000	0000	0010	1000	0001	1000
1000	0110	1000	0000	0100	0000	0000	0010
1100	0110	0010	0000	0010	1000	0001	1100
1000	0001	1100	0011	1110	0000	0000	0100
0000	0000	0000	0000	0000	0000	0000	1111
0000	0000	0000	0000	0000	0000	0000	1001
0000	0000	0000	0000	0000	0000	0000	0000

27. Compare la máquina de Von Neumann, el modelo IAS y la arquitectura ARC:
    - Memoria: capacidad (tamaño) de esta, rango de direcciones, formato de las palabras.
    - CPU: cantidad de registros, tamaño de estos.
    - Otras diferencias (formato y tipo de instrucciones, ¿RISC o CISC?, ¿cómo y qué instrucciones acceden a memoria?, etc).