Trabajo Práctico Nº 5
ARQUITECTURA DEL SET DE INSTRUCCIONES

Memoria

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?

2.- Un programa compilado para una arquitectura SPARC escribe en un archivo el entero sin signo de 32 bits 0xABCDEF01 y lo recupera correctamente. El mismo programa compilado para una arquitectura x86 (Pentium) también funciona correctamente. Sin embargo, cuando se transfiere el archivo entre máquinas, el programa de la computadora basada en Pentium lee incorrectamente el valor del archivo. ¿Cuál es el problema? ¿Cuál es el valor leído?

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.

Código Assembler

Declaración de variables y pasaje de parámetros de subrutina

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.

5.- Los tres programas siguientes tienen la misma función. Ordenarlos en función de los tiempos de ejecución esperados.

```
.begin
  value .equ 25
  ld      %r14,%r2
  add     %r0, value,%r1
  add     %r2,%r1,%r2
  st      %r2,%r14
  jmpl    %r15, 4, %r0
.end
```

```
.begin
  ld    %r14,%r2
  ld    [value], %r1
  add %r2,%r1,%r2
  st    %r2,%r14
  jmpl %r15, 4, %r0
  value:  25
.end
```

```
.begin
  ld      %r14,%r2
  add     %r0, 25,%r1
  add     %r2,%r1,%r2
  st      %r2,%r14
  jmpl    %r15, 4, %r0
.end
```

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.

7.- Idem el ejercicio anterior pero declarando modulo principal y subrutina en diferentes módulos.

8.- Idem ejercicio 6 pero en vez declarar una subrutina realizar la suma utilizando una macro y pasaje de parámetros por registro.
Comparar macros y subrutinas con respecto a (a) significado de los parámetros (b) espacio de memoria ocupada por el código de máquina (b) velocidad de ejecución.

Operaciones aritméticas

9.- Proponer una subrutina que recibe por stack un número entero en complemento a 2 y devuelve su valor absoluto.

10.- Escribir código que recibe por la pila dos números en complemento a 2 y determina si su suma es representable en 32 bits o no. En el primer caso devuelve un 0 y en caso contrario un -1.

11.- Idem anterior pero considerando que recibe números enteros sin signo.

12.- La multiplicación no forma parte de las operaciones aritméticas incluidas en el set de instrucciones ARC. Escriba una subrutina que recibe por la pila los números a multiplicar y devuelve por la pila su producto (la multiplicación no necesariamente debe ser implementada siguiendo un método óptimo). En caso de que el producto de ambos números no pueda ser representado en 32 bits, la subrutina devuelve -1.

13.- Proponer una rutina que recibe un número entero en el rango de 1 a 10 y devuelve su factorial o bien devuelve -1 en caso de que el número recibido se encuentre fuera del rango indicado.

14.- Proponer una rutina que recibe un número entero y devuelve el cuadrado del mismo. En caso de que el resultado no pueda ser representado con 32 bits debe devolver -1.

Operaciones con bits

15.- Escribir código que recibe a través de %r10 un número en punto flotante y devuelve su valor absoluto (también en punto flotante).

16.- Escribir un programa que recibe a través de la pila dos números en punto flotante simple precisión, los compara y devuelve respectivamente un 1 o un 2 según sea mayor el primero o el segundo.

17.- Un programa recibe a través de %r20 una palabra de 32 bits que representa dos números en complemento a 2 en sus dos bytes más significativos y en sus dos bytes menos significativos respectivamente. Proponer un código para ese programa tal que devuelva (también por %r20) la suma de ambos valores recibidos. Considere el manejo de condiciones de fuera de rango si lo considera necesario.

18.- Escribir una rutina que lee del stack un valor de 32 bits, cuenta la cantidad de 1´s comprendidos en su representación binaria y devuelve ese valor por stack.

Variables estructuradas

19.- Escribir un programa que determine la cantidad total de 1`s comprendidos en las representaciones binarias de todos los valores guardados en un arreglo cuyo largo y dirección de inicio son conocidos (estos llegan a través de la pila). Se espera que el código invoque la subrutina escrita como respuesta al punto 18.

20.- Un programa recibe via stack la dirección de inicio y el largo de un vector y devuelve su valor máximo y su valor mínimo también por stack. El cálculo del máximo se realiza invocando una subrutina declarada en el mismo módulo mientras que el cálculo del mínimo se realiza invocando una subrutina declarada en un módulo externo. Proponer un código para el programa principal y para ambas rutinas.

21.- Un programa declara un array de 30 elementos de 32 bits y le pasa a una subrutina su largo y localización en memoria. Esta subrutina determina si la suma de todos los elementos del array es un número par (devuelve un 1) o impar (devuelve un 0). En caso de que la suma excediese el rango de representación del sistema la rutina devuelve -1.
El programa debe inicializar a FFFFh todos los elementos del array en caso de determinar que su suma sea par o bien poner todos en cero si su suma excede el rango de representación.
Proponer un código para el programa principal y para la subrutina considerando las variantes de que ambos están declarados en el mismo módulo o en módulos diferentes.

Acceso a periféricos

22.- Un dispositivo mide simultáneamente 8 valores de temperatura y los almacena en ocho registros de 32 bits c/u los cuales se encuentran accesibles en el mapa de memoria de una computadora ARC mapeados a partir de la dirección B0101002h. Escribir una rutina que lea estos ocho valores y devuelva por stack el mayor de ellos.

23.- Un dispositivo de medición esta conectado a una computadora ARC y puede ser controlado a través de dos de sus registros, los cuales están mapeados en las direcciones de memoria D1=90001D00h y D2=90001D04h. El byte menos significativo de D1 refleja la medición de temperaturas realizada por el dispositivo (en el rango de 0 a 255 °C). El bit31 de D2 permite controlar el encendido de un indicador led de ese equipo (un 1 lo enciende y un 0 lo apaga). Escribir un programa que encienda el led sólo en caso de que la temperatura supere los 80 °C.
Al escribir el bit31 de D2 no deben alterarse el resto de los bits de esa posición.

24.- El ejercicio 36 de la tira de problemas correspondiente a Algebra de Boole plantea un sistema de control que responde a la siguiente consigna:

"El sistema de control habilita la entrada de materia prima si la temperatura del horno es mayor que 300 ^{o}C y la presión es inferior a 10 atmósferas, o si se llega a la mínima concentración de sales con una temperatura mayor que 300 ^{o}C, o si, siendo la presión mayor o igual que 10 Atmósferas, no hay suficiente concentración de sales."

Diseñe un programa que implemente esta lógica en código ARC considerando que el sistema es controlado por una computadora en la cual están mapeadas las siguientes entradas/salidas:

- las lecturas de presión y temperatura están mapeadas 81000000h y 81000004h.
- la insuficiencia de sales es representada por un 1 en el bit más significativo de 8100008h.
- la entrada de materia prima se habilita con un 1 en el bit0 de la posición 81000008h.

Código de Máquina

25.- Las instrucciones de salto condicional dirigen el flujo de programa hacia una instrucción identificada por una etiqueta. Indicar el rango máximo del salto hacia atrás y hacia delante medido en cantidad de instrucciones y en cantidad de bytes.

26.- La instrucción 'sethi' asigna una constante a los 22 bits más significativos de un registro. Sin embargo, sería útil que sethi permitiera asignar valores en los 32 bits del registro. Justifique porqué esto último no resulta posible.

27.- Proponga tres técnicas alternativas para cargar un registro con la constante 1000 y tres para cargar la constante 50000. Debido al formato del código de máquina, no todas deberían ser factibles en ambos casos. Indique cuáles sí y cuáles no.

28.- Las siguientes líneas de código incluyen dos errores. Identifíquelos y proponga formas alternativas para solucionarlos

```
.begin
Dir     .equ 10000
        .org 2048
main:  add  %r0,Dir,%r1
        ld   [array+4], %r2
        st    %r2,%r1
        jmpl  %r15+4,%r0

        .org A1010002h
array: .dwb 6
.end
```

29.- A continuación se presentan dos secuencias de código escrito en lenguaje C que hacen lo mismo: calcular la sumatoria de i desde 0 hasta n. El de la izquierda sigue un método recursivo mientras que el de la derecha sigue un procedimiento iterativo.

algoritmo recursivo	algoritmo iterativo

`int sumatorio (int n, int acumulado)`
`{`
`  if (n>0) return sumatorio (n - 1, acumulado + n);`
`  else return acumulado;`
`}`
	`int sumatorio (int n)`
	`{`
	`  int i, acumulado = 0;`
	`  for (i = n; i > 0; i--)`
	`  {`
	`  acumulado = acumulado + n;`
	`  }`
	`  return acumulado;`
	`}`

$\sum_{i=0}^{n} i = \sum_{i=0}^{n-1} i + n$	$\sum_{i=0}^{n} i = 0+1+2+\cdots+n$

Aunque ambas secuencias funcionan correctamente y generan el mismo resultado, la iterativa tiene un mejor rendimiento. ¿Por qué? Describa cómo quedan las instrucciones en Assembler y cuál es su comportamiento en tiempo de ejecución.

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.