Apellido y Nombre: [redacted]

email: [redacted]

Padrón: [redacted]

Fecha: ***12-dic-2019***

Cuatr. cursada: 2019 2^do Cuatri

Turno de TP: Viernes - Buzón

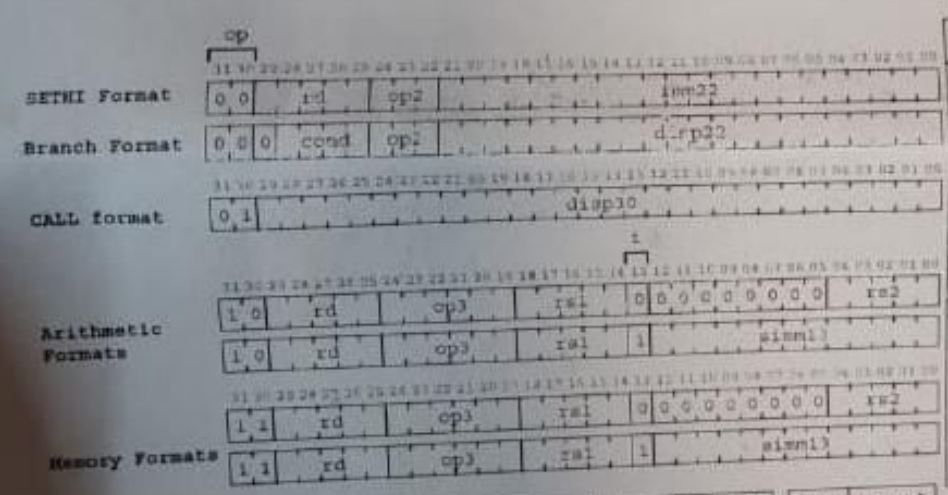
---

1) (a) Explique en qué aspectos se afectaría la operación del microprocesador ARC si una falla de hardware hiciera que la salida del flipflop menos significativo del registro de microinstrucciones quedara conectada permanentemente a cero. Sea específico y justifique detalladamente su respuesta en todos los pasos del ciclo de fetch.
   (b) Describa una estructura del tipo bus, indicar su propósito y sus aplicaciones específicas en el microprocesador y en el conjunto de una computadora.
   (c) Explique qué entiende por nanoprogramación.

2) Un periférico mapeado en la dirección A500A120h entrega una palabra de 32 bits cuyos 18 bits menos significativos contiene la información de interés. Escribir programa que declara un arreglo de 20 elementos y lo completa con los primeros 20 valores leídos del periférico mencionado (sus primeros 18 bits extendiendo el signo). Implementarlo:
   (a) declarando en el mismo módulo una rutina cuya función es leer el periférico mencionado, no recibe parámetros de entrada y devuelve por stack el valor de 18 bits extendido en signo a 32.
   (b) declarando una macro que cumple una función similar a lo indicado en (a)

3) Compare el linkeado en tiempo de carga con el que ocurre cuando se utilizan librerías de linkeado dinámico. Detalle los inconvenientes que intenta solucionar cada uno de estos sistemas.

4) Explique detalladamente qué se entiende por fallo de cache. ¿Por qué razones es deseable que no ocurra? ¿Conoce pautas de programación en lenguajes de alto nivel para minimizar su ocurrencia?



F_3 F_2 F_1 F_0	Operation
0 0 0 0	ANDCC (A, B)
0 0 0 1	ORCC (A, B)
0 0 1 0	XORCC (A, B)
0 0 1 1	ADDCC (A, B)
0 1 0 0	SHL (A, B)
0 1 0 1	AND (A, B)
0 1 1 0	OR (A, B)
0 1 1 1	NOR (A, B)
1 0 0 0	ADD (A, B)
1 0 0 1	LSHIFT2 (A)
1 0 1 0	LSHIFT10 (A)
1 0 1 1	SIMM13 (A)
1 1 0 0	SEXT13 (A)
1 1 0 1	INC (A)
1 1 1 0	INCPC (A)
1 1 1 1	RSHIFTS (A)

op	Format
00	SETHI/Branch
01	CALL
10	Arithmetic
11	Memory

op2	Inst.
010	branch
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	branch
0001	be
0101	bcs
0110	bneg
0111	bvs
1000	ba

cond	Operation
0 0 0	Use NEXT ADDR
0 0 1	Use JUMP ADDR if I = 1
0 1 0	Use JUMP ADDR if C = 1
0 1 1	Use JUMP ADDR if Z = 1
1 0 0	Use JUMP ADDR if N = 1
1 0 1	Use JUMP ADDR if (Z OR (N XOR V)) = 1
1 1 0	Use JUMP ADDR
1 1 1	DECODE

Final 13-12-2018

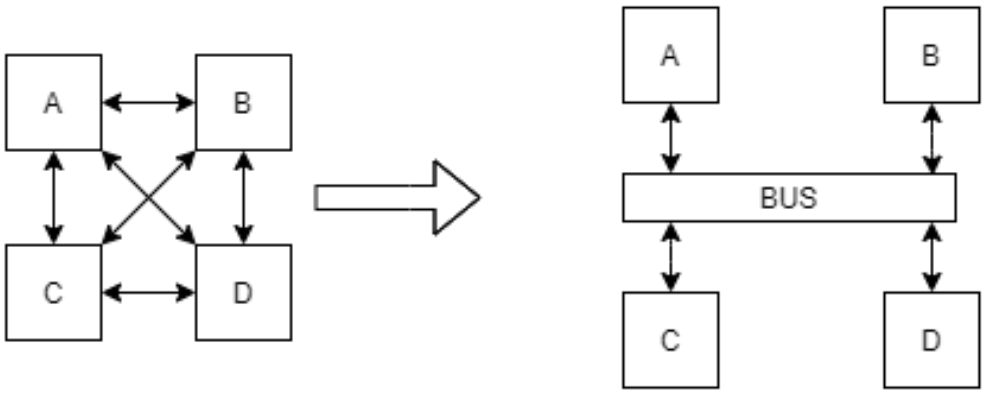
1. Ejercicio 1

   a) Los pasos del ciclo fetch son:

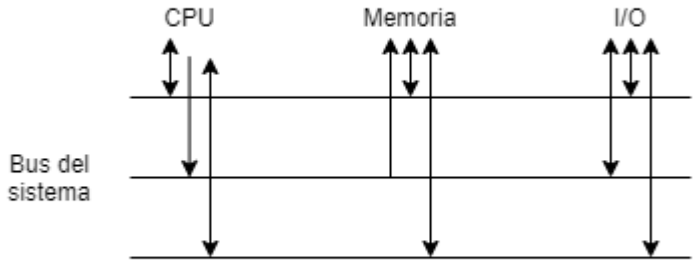
   - Búsqueda de la próxima instrucción
   - Decodificación del código
   - Búsqueda de operandos en memoria(si los hubiera)
   - Ejecución y almacenamiento
   - Volver al paso 1

   Paso 1: Búsqueda de la próxima instrucción:
   Para este paso la falla no afectaría. Esto es porque la microinstrucción llevada a cabo es la de la línea 0(`R[IR] <-- AND(R[PC], R[PC]); READ;`)la cual puede ser llevada a cabo independientemente de la falla..
   Paso 2: Decodificación del código:
   En este paso, se decodifica la instrucción utilizando sus campos OP y OP3/OP2. En este paso, la falla tampoco afectaría.
   Paso 3: Búsqueda de operandos en memora:
   Paso 4: Ejecución y almacenamiento:
   En este paso empieza a afectar la falla. En particular, la falla afectará en que el campo JUMP ADRESS siempre va a ser un número par(último bit=0). Puede existir una microinstrucción cuya dirección de salto sea impar y esta se vería afectada. Si por ejemplo la microinstrucción tuviese la siguiente línea:
   `IF R[IR[13]] THEN GOTO 1761`
   Si el bit 13 del IR fuese 1, esto significaría que la próxima microinstrucción a ejecutar sería la que se encuentra en la línea 1761, pero por el error mencionado en el enunciado se saltaría a la dirección 1760 produciendo un resultado no deseado el cual podría ser por ejemplo un loop infinito o acceso a zonas prohibidas.
   Paso 4: Volver al paso 1:
   En el último paso de la microinstrucción se suele volver a la línea 2047 con un GOTO 2047. En este caso, por la falla no se podría saltar a esta línea no sería posible dado que es impar y se saltaría, en cambio, a la línea 2046. Para otras microinstrucciones(por ej JMPL) que en la última línea hacen un GOTO 0, no afectaría la falla del hardware.

   b) El bus es una solución eficiente para sistemas en donde muchos módulos se interconectan entre si. El uso del bus reduce la cantidad de cableado requerido para las interconexiones.



Las ventajas del bus son básicamente que el diseño es mas simple y de menor área. Algunas desventajas son que no es posible conectar todo al mismo tiempo y que requiere de un control que habilite y deshabilite las entradas y salidas.
Algunas aplicaciones especificas en el microprocesador y el conjunto de la computadora son por ejemplo: El bus A(bus de direcciones a memoria), el bus B(bus de datos a memoria) y el bus C(bus de datos desde memoria).



c) La nanoprogramación es una técnica para ahorrar espacio de memoria. Se puede ahorrar espacio de memoria de microprograma colocando una copia de cada palabra de microcódigo en un elemento de nanoalmacenamiento, usando la memoria de microprograma como índice a la memoria de nanocódigo.
En lugar de acceder solo a microcódigo ahora se debe acceder al microcódigo y luego al nanocódigo. La máquina funcionará mas lentamente pero ocupando

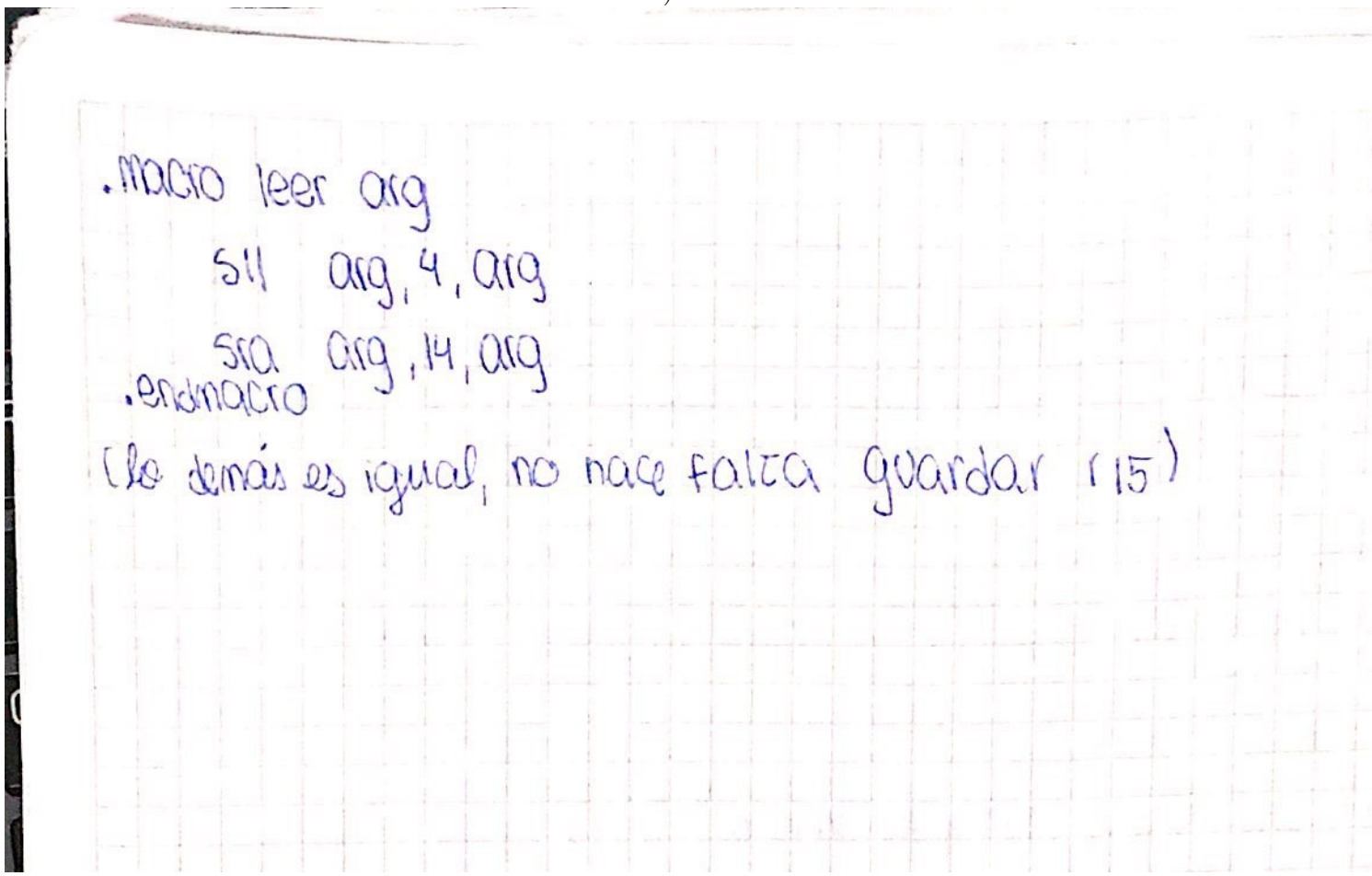
menos espacio.

2. Ejercicio 2

   a)

```
.begin
.org 2048

.macro Posm arg
        Add %r14, -4, %r14
        st  arg, %r14
.endmacro
.macro Pop arg
        ld  %r14, arg
        Add %r14, 4, %r14
.endmacro

DIR .equ 3000

Add  %r15, %r0, %r16     ! me guardo la ref a r15 en r16

add  %r0, DIR, %r1       ! Puntero al arreglo

Setni 0500a0h, %r20

sll  %r20, 2, %r20

Add  %r20, 120h, %r20    !Cargo el Perif

add  %r0, 0, %r2         !r2 long del arreglo

loop: anocc %r2, %r2, %r0
      be    end
      call  leer
      Pop   %r5          ! guardo en r5 lo que retorna la rutina
      st    %r5,%r1      ! guardo en el array el res
      Add   %r1, 4, %r1  ! avanzo array
      sub   %r2,1, %r2
      ba    loop
salida: jmpl  %r16,4, %r0

leer:  ld  %r20, %r3     ! Cargo lo leido en r3
       sll %r3, 14, %r3  ! saco los bits que no
       sra %r3,14 , %r3  ! quiero
       Posm %r3          ! dejo en el tp Pila el res
       jmpl %r15, 4, %r0
end: ba salida

.org 3000
arreglo: .dwb 20
.end
```

b)



3. Ejercicio 3
   El linkeado en tiempo de carga carga todos los módulos que incluye el programa para ejecutar. El linkeado dinámico solamente carga los módulos utilizados. Una ventaja de DLL es que el programa inicial es mas liviano y usa menos recursos porque si hay un módulo muy grande que no se usa no lo carga. Una desventaja es que es frágil porque el programa en ejecución no puede resolver que hacer si no encuentra la DLL a ser cargada.

4. Ejercicio 4
   La memoria caché existe para ahorrar tiempo de ejecución reduciendo accesos a memoria principal. Lo que hace esta memoria es almacenar los programas mas usados, entonces antes de acceder a memoria principal se accede a memoria caché para ver si está ahí, en caso de no estar ahí se accede a memoria. Esto es porque solo se usa un 10& de la memo principal en general entonces los datos más usados se guardan en la memoria caché para que sean accesibles.
   El fallo de caché sucede cuando algún dato se va a buscar a la memoria cache y este no se encuentra ahí. Se busca evitar el fallo porque implica acceder a memoria principal que es justamente lo que se está tratando de evitar.

Algunas formas de evitar el fallo de la memoria caché es declarar int separados de largos bloques como arrays para que queden en el mismo bloque que la caché.