```
--------------------------------------------------------------------------------
Ejercicio 1
--------------------------------------------------------------------------------
------------
Modulo 1
------------

.begin
.org 2048

.macro push arg
        add %r14,-4,%r14
        st arg,%r14
.endmacro

.macro pop arg
        ld %r14,arg
        add %r14,4,%r14
.endmacro

.extern complemento_a_dos

!cargo la direccion de memoria

A .equ B2000h
B .equ 000h

sethi A,%r2
sll %r2,2,%r2
add %r2,B,%r2           !en r2 esta la direccion de memoria

add %r0,1,%r1   !en r1 guardo la mascara para ver si es impar

ld %r2,%r3              !contenido de la dir de memoria almacenado en %r3
and %r3,%r1,%r4 !en r4 almaceno el resultado

add %r0,4,%r8           !guardo el i para multiplicar

add %r15,0,%r16 !back up del %r15

push %r3                !cargo en la pila el numero leido

addcc %r4,-1,%r0
bneg es_par
be es_impar

pop %r13

st %r3,%r2
halt

es_impar:       call complemento_a_dos
                jmpl %r16+4,%r0
```

```
es_par:         call multi_por_cuatro
                jmpl %r16+4,%r0



multi_por_cuatro:       pop %r24
                        add %r15,0,%r17 !guardo dir para salir del
multi_por_cuatro
                        add %r0,%r0,%r5 !almaceno el resultado de mi multi
                        call multiplicar
                        push %r6
                        jmpl %r17,4,%r0

multiplicar:            andcc %r8,%r8,%r0
                        be fin_multiplicacion
                        add %r24,%r6,%r6
                        add %r8,-1,%r8
                        ba multiplicar

fin_multiplicacion:     jmpl %r15+4,%r0

.end
```

Modulo 2

```
.begin
.org 2048

.global complemento_a_dos

complemento_a_dos:      pop %r24
                        orncc %r1,%r1,%r1
                        push %r1
                        jmpl %r15,4,%r0

.end
```

Ejercicio 2

Escribir tabla de simbolos del codigo anterior

-¿De que manera es utilizada en tiempo de ejecucion?-

La tabla de simbolos es generada en la primera pasada del ensamblador, teniendo asi la ventaja de referencia previa el ensamblador. Es por esto, que en la segunda pasada cuando el ensamblador comienza a generar el codigo de maquina, ya teniendo esta tabla de simbolos evita un error de emsamblado, ya que ya conoce todos los respetivos valores a cada simbolo.
Y si son globales, externos y/o reubicables. Por lo tanto cuando el linker tenga que unificar en este caso los dos modulos, y resolver las referencias externas o globales, y realizar las reubicaciones necesarias, podra utilizar esta tabla de simbolos para obtener toda la informacion que ésta le provee.

Ejercicio 3

(Item A)
Ciclo de fetch o de busqueda-ejecucion
1. SBúsqueda en memoria de la próxima instrucción a ser ejecutada.
2. Se la decodifica
3. Se busca los operandos en memoria si los hubiera
4. Se ejecuta y se almacenan los resultados
5. Volver al primer paso.

Por lo tanto, se busca en memoria la instruccion que se quiere ejecutar. Con el formato del IR se realiza la decodificacion de la instruccion. De esta forma sabremos a que direccion se encuentra la microinstruiccion que debemos ejecutar. El campo del MIR, contiene los campos i,iMux (siendo i={A,B,C}), Read, write, alu, cond y jump addr. En los campos A y B, iran los operandos de la funcion que le diremos a la ALU que realice, en C donde se almacenara el resultado. Los campos Amux,Bmux y Cmux, indicaran si el operando proviene del MIR o del IR. El read y el write, pueden ser ambos ceros, o uno en el caso de que se lea o escriba respectivamente en memoria. Nunca pueden ser ambos 1 a la vez. En el campo de la ALU le idicaremos que operacion realizar. Cond que tipo de salto debe dar y jump adrre la direccion a la que debe saltar (excepto que sea NEXT o DECODE el cond, en esos dos cosas el campo de jump addr se ve ignorado.
Luego se va a distribuir la informacion, de manera que los multiplexores recibirán una direccion del MIR y otra del IR, y a su vez del iMux del MIR que les indicara que direccion deben leer, si la que proviene del mir o del ir. Una vez seleccionada la direccion, se la comunicara el multiplexor al decodificador que sera el encaragado de encender el registro que se va a utilizar en cada bus y apagar los restantes. Es decir si en el bus B debe estar el registro 5, el decodificador pondra el bit 5 en uno y todos los restantes en cero.
A travez de los bus, viajara la informacion a la ALU, que tambien recibirá la funcion desde el mir. Luego de realizar la operacion, via el bus C depositara el resultado, a su vez, le enviara a la logica de control de saltos los flags.
La logica de control de saltos al recibir los flags, la condicion de salto (desde el mir) y el bit13 desde el IR, enviará al CS Address Mux, dos bits, indicando si la proxima direccion a ejecutarse sera la que esta en el campo del Jump Addr del mir, o si se debe incrementar la que se estaba ejecutando o si se debe realizar una decodificacion. De esta manera, se le informa a la memoria de microinstrucciones, la cual debe buscar la microinstruiccion que se le indique.

Agregar micros (tengo q ver cuales son) y agregar circuito masomeno.

Ejercicio 4
