FINAL Franco

Ej ①

Un programa recibe por STACK la dirección de inicio y largo de un arreglo. Se sabe que este arreglo estará conformado por elementos en PUNTO FIJO con 4 bits para la parte fraccionaria. El programa debe sumar todos los elementos del arreglo que tengan parte fraccionaria nula y escribir el resultado en un periférico mapeado en la dirección ABC0232h. Para verificar si cada elemento cumple con la condición indicada se debe pasar el mismo a una rutina por vía STACK devolviendo esta también por STACK un 1 si la condición se cumple o un 0 si no se cumple. Subrutina y programa principal están en el mismo módulo.

```
.begin
.org 2048

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

.macro push reg
       add %r14, -4, %r14
       ST reg, %r14
.endmacro
```

```
A .equ   ABC00 h
B .equ   232 h
SETHI   A , %r1
SLL     %r1, 2, %r1
add     %r1, B, %r1      ! en %r1 almaceno la dir del periferico

pop %r2                  ! almaceno el largo del arreglo
pop %r3                  ! almaceno la direccion del array

and %r0, %r0, %r4        ! %r4 será mi acumulador
 call FOR
 St %r4, %r1

FOR :  addcc %r2, -4, %r2
       bneg FIN
       add %r2, %r3, %r5     ! almaceno la pos. del array
       ld %r5, %r6           ! guardo en %r6 el contenido de %r5
       add %r15, 0, %r20     ! backup para salir del FOR
       push %r6
       call es_valido
       pop %r7               ! guarda el 100
       addcc %r7, -1, %r0
       be sumar
       ba FOR
```

```
                pop %r6
es_vaudo:       ld %r6, %r10    ! back up del contenido
                sll %r10, 28, %r10  ! me quedo solo con los 4 bits
                                    q me importan
                andcc %r10, %r0, %r0
                be son_ceros
                push uno
                jmpl %r15+4, %r0

son_ceros:      push %r0
                jmpl %r15+4, %r0

suma:           add %r4, %r6, %r4
                jmpl %r15+4, %r0

FIN:            jmpl %r20+4, %r0

uno: 1
.end
```

(a) Ej (2) Proponer un microcodigo para una INSTRUCCION call

INDICAR microinstrucción de la segunda linea

- R[15] ← and (R[pc], R[pc])
- R[tempo] ← add (R[ir], R[ir])
- R[tempo] ← add (R[tempo], R[tempo])
- R[pc] ← add (R[pc], R[tempo])
  Goto 0

ir es el R[=7]

La locación? el R[15] es el registro de enlace

Indicar valores de · R[tempo] ← add (R[ir], R[ir])

100101 0 100101 0 100001 0 00 1000 000 000000000

(b) Proponga un circuito para implementar el r.ro indicando claramente cada conexión con los demas componentes de la microarquitectura

El r.ro siempre contiene ceros, no puede modificarse, en la implementacion de estudiada el r.ro no tiene entradas del bus C, ni tampoco del decodificador C, por consiguiente, no es necesario el uso de FF.

c) Explique de que forma los flags de la ALU controlan el flujo del microprograma detallando todos los componentes circuitales que intervienen.

los FLAGS surgen de las operaciones de la ALU, indicando si hubo carry (c), overflow (v) y si el resultado es negativo (n) o cero (z). Los FLAGS serán enviados luego del procesamiento de cada microinstrucción por la ALU a la logica de control de saltos, quien contemplando los FLAGS, la cond de salto (MIR) y el bit 13, le indicará al CS Address MUX que microinstrucción debe ser la siguiente a ejecutar.

Ej ③

a) Comparar ensambladores de una y dos pasadas contemplando ventajas y desventajas

El ensamblador de una pasada va pasando a lenguaje de maquina cada linea de código que lee. En cuanto al ensamblador de dos pasadas, en la primer pasada va generar una tabla de simbolos (rotulos, variables, etiquetas), donde indicará si las mismas son ext, globales, reubicables y su dirección de memoria en el modulo, para entonces en su segunda pasada generará el codigo de maquina, insertando en el mismo los valores de los simbolos ya conocidos para ese momento.

La gran ventaja con la que cuenta el ensamblador de dos pasadas es la referencia previa en su segunda pasada, dado que los simbolos ya han sido definidos, lo q le permite saber de antemano si hubo redefiniciones y si hay un error de def lo detecte de 1 primer momento previo a generar el código de maquina. Por consiguiente, la gran desventaja del ensamblador de una única pasada es no contar con la tabla de simbolos, ya que esta va a ir generando codigo de maquina, por lo tanto si hubiera un problema de no def de una variable (por ej), lo sabrá luego de haber generado ya líneas de maquina.

NOTA

(b) Explique si en la programación con lenguajes de alto nivel (y exclusivamente a los fines de una mayor velocidad de ejecución) es más conveniente o menos conveniente:

(a) Definir los valores constantes declarandolos como constantes

(b) Declarandolos como una variable inicializada al correspondiente valor

Indique las ventajas y desventajas, y las limitaciones en cada caso analizandolo sobre la base del assembler que se genera con cada una

En tiempo de ejecución resulta más conveniente definir los valores constantes declarandolos en el código como constantes, aunque tiene la limitación del tamaño de constante posible a cargar ([13/22] bits ???), pero resulta más veloz que la segunda opción. Dado que declarando como una variable inicializada al correspondiente valor, cuenta con la desventaja que dichas variables son cargadas a memoria, y el acceso a memoria genera que la ejecución del programa sea más lenta.

(4)

(a) El hardware de cache maneja información sobre la base de una estructura de tipo tabla. Indique la información guardada en dicha tabla, y de que manera se organiza el funcionamiento del cache sobre esta base.

La memoria caché es una tabla formada por bloques de direcciones (una cant mucho menor a las dir de RAM). El almacenamiento en estos bloques dependen de el principio de localidad. Este principio se divide en localidad temporal y localidad espacial.

Refiriendose a localidad temporal a que cuando un programa hace ref a una locación de memoria, es muy probable que en un corto plazo vuelva a acceder a ella y la locación espacial, tras acceder a una locación dada es mucho mas probable acceder a pos cercanas a ella que a pos lejanas.

En cuanto a funcionamiento, una vez que se solicite una variable, en primer lugar se verifica si la misma no esta en caché. Si esta, se lee desde ahi y no se accede a RAM. Si no está, se genera una pos vacia en caché, donde se va a copiar el bloque de RAM que esté almacenando a la variable. (de esta forma estamos cumpliendo con el ppio de localidad)

(b) ¿Qué objetivos persigue la técnica de bancos entrelazados en memoria RAM?

Los objetivos que persigue la técnica de bancos entrelazados en memoria RAM son: enmascarar el tiempo de refresco (para no perder tiempo, cada vez que leo o escribo, refresco el resto de los bits) Además, mejora el rendimiento, dado que no pierdo tiempo si es que deseo ingresar a direcciones de memorias consecutivas.