Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2343 – Arquitectura de Computadores (I/2013)

Solución Examen

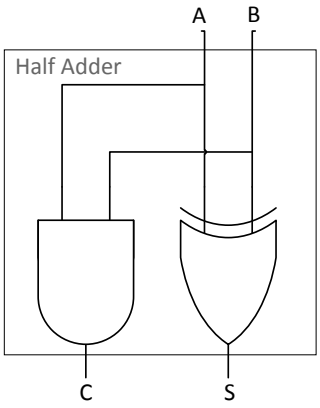
Pregunta 1

a) ¿Cómo se puede calcular la cantidad de bits necesarios para representar a los elementos de un conjunto acotado de números enteros? **(1 pto.)**
Solución: Se necesita tomar el número con mayor valor absoluto del conjuto, τ, y ver cuanto bits son necesarios para representar este valor absoluto. Luego, dado que estamos trabajando con números enteros, se agrega un bit más para poder utilizar el complemento a 2. La expresión final para la cantidad de bits es entonces: : $2 + \lfloor \log_2 \mid \tau \mid \rfloor$. En caso que el número con mayor valor absoluto sea 0, sólo se necesita un bit.

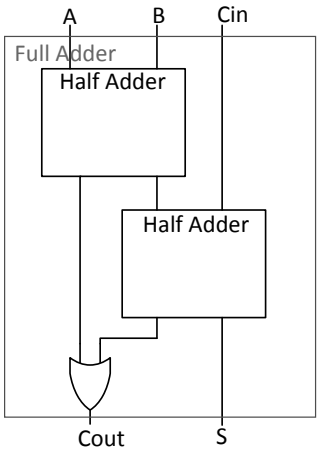
b) Explique el concepto de endianness en memoria. ¿Es necesario especificar el endianness cuando se lee desde un registro? **(1 pto.)**
Solución: El endianness indica el orden en que se almacenan los bytes de un dato en una memoria. Esto es imprescindible, ya que en las memorias, cada direccion indica una palabra (byte). En los registros no es necesario definir endianness, ya que los valores se almacenan como un todo y el acceso es al registro completo.

c) ¿Es posible eliminar completamente el problema con los números infinitos usando una representación de punto flotante? **(1 pto.)**
Solución: No es posible eliminar este problema, ya que es inherente a las representaciones posicionales y depende de la base. Es posible aplacarlo usando formatos numéricos avanzados, pero nunca se puede eliminar del todo.

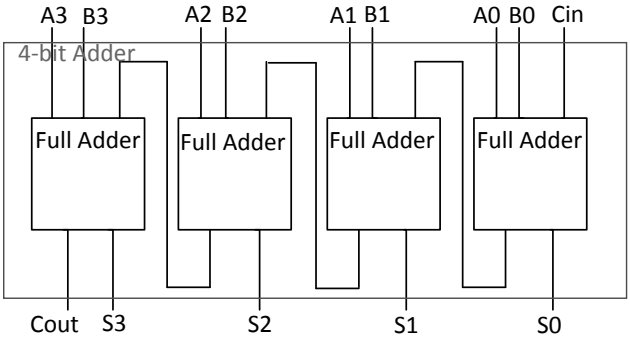
d) Diseñe en detalle, a partir de compuertas básicas, un sumador/restador de 4 bits. **(3 ptos.)**
Solución: El primer paso corresponde al diseño de un half-adder de 1-bit:



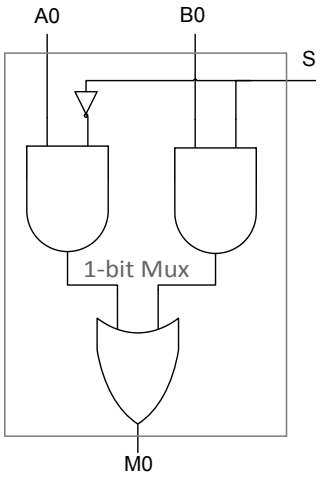
Luego, usando half-adders, diseñamos un full-adder:

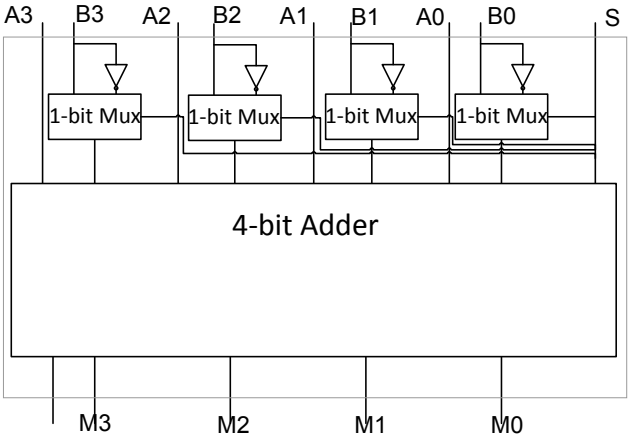


Usando 4 full-adders, se construye de manera trivial un sumador de 4 bits:



Finalmente, diseñamos un mux de 1 bit y junto con un sólo adder y aprovechando al relación entre complemento a 2 y la negación de bits, construimos el sumador/restador de 4 bits:


A0
B0
S
1-bit Mux
M0


A3
B3
A2
B2
A1
B1
A0
B0
S
1-bit Mux
1-bit Mux
1-bit Mux
1-bit Mux
4-bit Adder
M3
M2
M1
M0

Pregunta 2

a) ¿Que modificación realizaría al computador básico para reducir el número de instrucciones de los programas? Describa brevemente como lo implementaría. **(1 pto.)**
Solución: Una modificación sencilla y muy efectiva es agregar a la ALU unidades de multiplicación y división. De esta manera, cada vez que se necesite realizar una de estas operaciones, se necesitará sólo una instrucción, a diferencia del caso actual, en donde se necesita realizar un loop.

b) ¿Qué es stdcall y por qué es necesario que exista? **(1 ptos.)**
Solución: stdcall es una de las muchas convenciones de llamada a subrutina existentes. Son necesarias para específicar sin ambigüedades la manera en que se entregan y reciben datos desde una subrutina.

c) Escriba en assembly x86 un programa que encuentre la moda de un arreglo de número naturales. El tamaño del arreglo es arbitrario y en caso de haber mútiples modas, se debe retornar la de menor valor. **(4 ptos.)**
Solución:

```
        MOV CX, 0
        MOV SI, 0
        start:
        CMP SI, n
        JE end
        LEA BX, array
        MOV CL, [BX + SI]   ;CL almacena el valor a analizar
        MOV DI, CX
        SHL DI, 1           ;los elementos de freq son de tipo dw
        LEA BX, freq
        MOV AX, [BX + DI]   ;esta linea y las dos siguientes
        ADD AX, 1           ;actualizan en freq las apariciones
        MOV [BX + DI], AX   ;del valor almancenado en CL
        CMP AX, mode_count  ;si las apariciones son mayores que
        JL next             ;las de la moda actual, se compara su valor
        CMP CL, mode        ;si el valor en CL es menor que la moda
        JG next             ;actual y tambien tiene mas apariciones, se
        MOV mode, CL        ;actualiza la informacion en las variables
        MOV mode_count, AX  ;mode y mode_count
        next:
        ADD SI, 1
        JMP start
        end:

        mode        db    127
        mode_count  dw    0
        n           dw    ?
        array       db    ?,?,...
        freq        dw    0,0,...
```

Pregunta 3

a) ¿Cuál es la función del controlador de DMA? Describa en detalle con qué elementos se conecta y comunica. **(1 pto.)**
Solución: El controlador de DMA se encarga de realizar transferencias de datos entre la memoria y dispositivos de I/O mapeados a memoria. Se comunica con la memoria, los dispositivos y con la CPU. Debe estar conectado al menos al bus de I/O (datos), al de direcciones y al controlador de interrupciones o la CPU.

b) Describa en detalle y compare las interrupciones por software y las excepciones. **(1 pto.)**
Solución: Una interrupción de software es una interrupción generada mediante el llamado, por parte de un programa, a una instrucción especial (INT) que a su vez llama a la ISR correspondiente. Por otra lado, una excepción es generada por la CPU cuando detecta una evento especial o un error que no puede o no debe ser manejado por el programa.

c) El tamaño de páginas/marcos es un parámetro esencial para tener un buen rendimiento. ¿Cómo fijaría este parámetro en la práctica? **(1 pto.)**
Solución: Una manera de fijar este parámetro es ejecutar una gran cantidad de programas, variando el tamaño de las páginas, y medir su rendimiento y cantidad de memoria utilizada como función del tamaño de las páginas. Luego, se utiliza el tamaño que presente la mejor combinación de estos valores.

d) ¿Cuál es la función de la TLB? ¿Es necesario reiniciarla cuando hay un cambio de contexto y en caso de ser necesario, como se puede evitar esto? Todas las respuestas deben ser justificadas en detalle. **(1 pto.)**
Solución: La TLB es una memoria caché muy pequeña utilizada para acelerar el funcionamiento de la MMU. Al almacenar la transformación página/marco y al ser esta diferente para cada programa, es necesario limpiarla en los cambios de contexto, a menos que se agregue a cada bloque de esta caché una columna que indique a que proceso pertenece la información almacenada. De esta manera, no es necesario borrar la TLB, ya que los bloques con un identificador de proceso diferente al actual será tomados como no validos.

e) En un computador, la memoria principal tiene 16 bloques y la memoria caché tiene sólo 4. La CPU ejecuta un programa que requiere de la memoria principal la primera palabra de los bloques 5, 7, 5, 4, 10, 5, 3, 10, 5, 4 en el orden señalado. Suponiendo que inicialmente la memoria caché está vacía y que la función de correspondencia de la memoria caché es de mapeo directo, indique cuál es la tasa de aciertos de la memoria caché. **(2 ptos.)**
Solución:

- 5 (Bloque 1) - Miss
- 7 (Bloque 3) - Miss
- 5 (Bloque 1) - Hit
- 4 (Bloque 0) - Miss
- 10 (Bloque 2) - Miss
- 5 (Bloque 1) - Hit
- 3 (Bloque 3) - Miss

- 10 (Bloque 2) - Hit
- 5 (Bloque 1) - Hit
- 4 (Bloque 0) - Hit

Luego, el hit rate es $\frac{5}{10} = 0{,}5$.

Pregunta 4

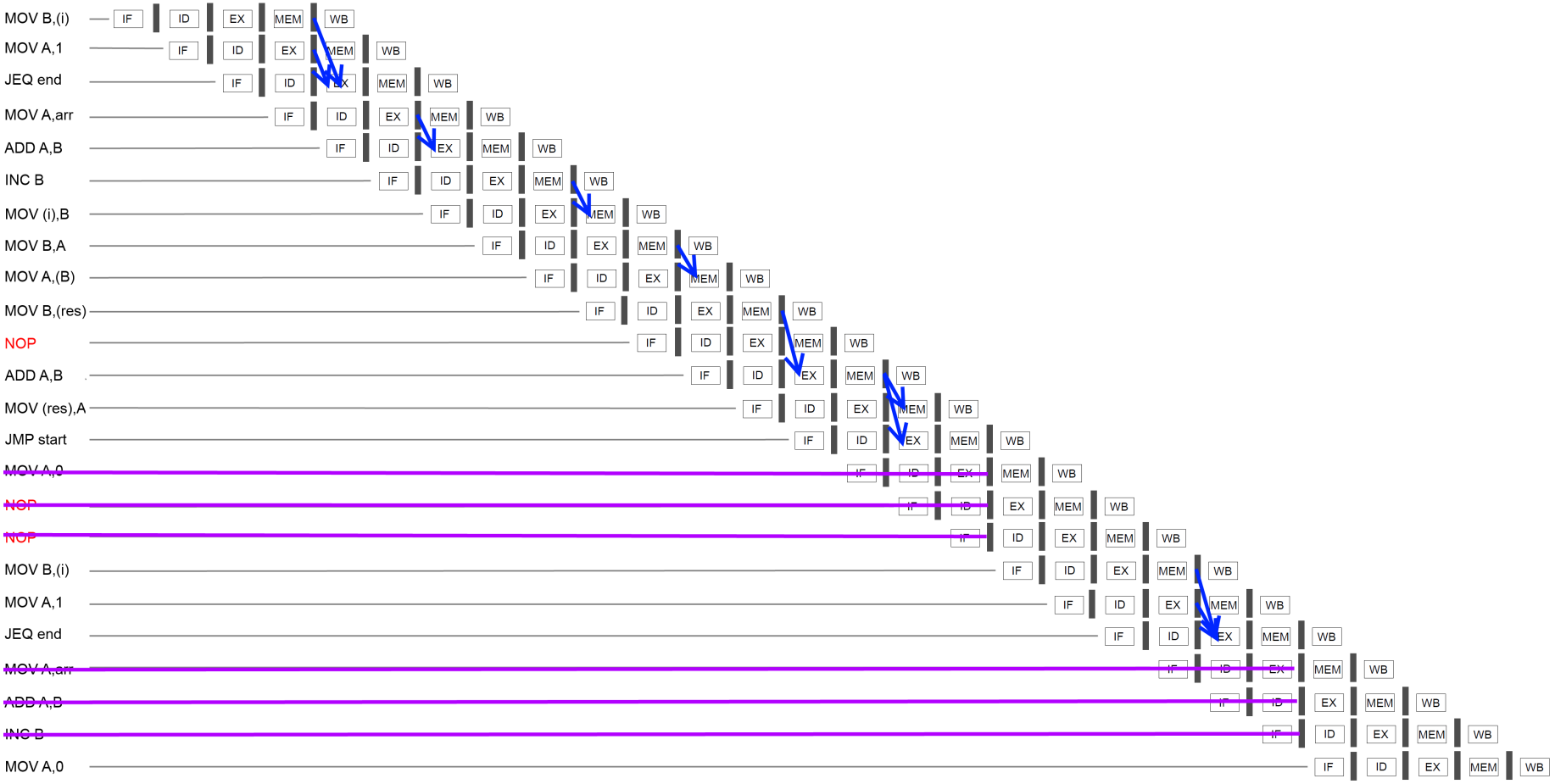
a) Indique qué capacidades/intrucciones gana y pierde el computador básico al agregar soporte para paralelismo a nivel de instrucción. **(1 pto.)**
Solución: Logicamente, el computador gana la capacidad de procesar multiples instrucciones en paralelo. Dentro de las características que se pierden está i) el llamado a subrutinas, ya que no hay SP ni conexión entre memoria y PC, i) los opcodes N, C y V, la conexión entre la salida de memoria y el MUX B, la entrada de datos de la memoria ahora sólo puede provenir de los registros A y B y no de la ALU y la dirección de la memoria ahora proviene de la ALU.

b) Un computador con microarquitectura avanzada posee un pipeline de 30 etapas. ¿Cuál es el elemento de hardware, sin contar los registros entre etapas, que tiene el mayor impacto (positivo y negativo) en el rendimiento ? Justifique su respuesta. **(1 pto.)**
Solución: Dado que se tiene un pipeline muy profundo, la unidad más importante será la de salto, más especificamente la predicción de saltos, ya que si está mal implementada, cada salto mal estimado tendrá un costo muy alto en ciclos, logicamente dependiendo de la profundidad a la que se encuentre la unidad de salto.

c) Determine el número de ciclos que se demora el siguiente código y la cantidad de ciclos perdidos, si se utiliza el computador básico con pipeline de 5 etapas. Indique además la cantidad de ciclos que hubiese tomado en un computador sin pipeline. Detalle en un diagrama los estados del pipeline por instrucción. El pipeline tiene forwarding entre todas sus etapas, el manejo de stalling es por software (instrucción NOP) y predicción de salto asumiendo que no ocurre. Indique en el diagrama cuando ocurre forwarding, stalling y flushing. **(4 ptos.)**

```
DATA
    arr     2
            3
    i       0
    res     0
CODE
    start:  MOV B, (i)
            MOV A, 1
            JEQ end
            MOV A, arr
            ADD A, B
            INC B
            MOV (i), B
            MOV B, A
            MOV A, (B)
            MOV B, (res)
            ADD A, B
            MOV (res), A
            JMP start
    end:    MOV A, 0
```

Solución: Basado en el siguiente diagrama, los ciclos totales que toma el proceso son 28, mientras que los ciclos perdidos son 7. Finalmente, sin usar un computador con pipeline, el proceso hubiera tomado 17 ciclos.


MOV B,(i)
MOV A,1
JEQ end
MOV A,arr
ADD A,B
INC B
MOV (i),B
MOV B,A
MOV A,(B)
MOV B,(res)
NOP
ADD A,B
MOV (res),A
JMP start
MOV A,0
NOP
NOP
MOV B,(i)
MOV A,1
JEQ end
MOV A,arr
ADD A,B
INC B
MOV A,0
IF
ID
EX
MEM
WB