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

IIC2343 — Arquitectura de Computadores — 1' 2022

Interrogación 1

Responde solo TRES de las siguientes cuatro preguntas.

1 Representación de números

1.1 Complemento de 2 (1 pts)

Dados dos números enteros, P y Q, en complemento de 2, queremos calcular la resta P – Q. ¿Bajo qué condiciones relativas sólo a los signos de P y Q, el valor de la resta no puede producir overflow? Justifica de manera clara y precisa tu respuesta, y ejemplifica tu justificación.

1.2 Punto flotante (2 pts)

Considere los siguientes números decimales,

$$n_1 = 320000.25_{10}$$

$$n_2 = 10.15625_{10}$$

(a) Escriba ambos números en su representación de punto flotante según el estándar IEEE-754 de precisión simple (32 bits). **(1 pts)**

(b) Sume ambos números de punto flotantes. Comente el resultado. **(1 pts)**

1.3 Hewlett-Packard (3 pts)

Los computadores HP 2114, 2115 y 2116 usaban un formato de representación numérica que utilizaba 32 bits para representar números, donde los 24 bits de la izquierda almacenaban la fracción en complemento de dos, y los 8 bits de la derecha almacenaban la representación del exponente, que se almacenaba utilizando representación de signo y magnitud, con el signo siendo el bit de más a la derecha. El 1 de la representación en notación científica no está oculto en esta representación. Escriba el patrón de bits para representar el valor -5.451×10^4. Además, comente como se compara en rango y precisión respecto al estándar IEEE-754.

2 Circuito combinacional

(a) Dibuje el diagrama de un circuito que tome como entrada $p = p_1p_0$ y $x = x_4x_3x_2x_1x_0$, donde la salida $y = y_4y_3y_2y_1y_0$ sea el número resultante después de realizar p veces *shift left* sobre x, y exista un bit f, que corresponde a un *flag* que indica si después de realizar la operación hubo cambio de signo, donde $f = 0$ indica que no hubo cambio, y $f = 1$ que si hubo cambio. **(2.4 pts)**

(b) Dibuje el diagrama de un circuito que tome como entrada $p = p_1p_0$ (número sin signo) y $x = x_4x_3x_2x_1x_0$, donde la salida $y = y_4y_3y_2y_1y_0$ sea el número resultante después de sumar x con p, y exista un bit f, que corresponde a un *flag* que indica si después de realizar la operación hubo overflow, donde $f = 0$ indica que no hubo overflow, y $f = 1$ que si hubo overflow. **(2.4 pts)**

(c) Ahora dibuje el diagrama de un circuito que tome como entrada $p = p_1p_0$, $x = x_4x_3x_2x_1x_0$ y $S = s0$, donde la salida $y = y_4y_3y_2y_1y_0$ y $f = f_0$ depende de S. Si $S = 0$, al número x se le deben realizar p veces *shift left*. Si $S = 1$, se debe sumar x con p. Considere que se tienen los circuitos de (a) y (b) correctos. **(1.2 pts)**

3 Arquitectura

Considera la configuración del computador básico que se adjunta. Como vimos, esta configuración permite manejar números enteros de 8 bits (el tamaño de los registros, el ancho de los buses, etc.). Para poder manejar números de 16 bits, tenemos que usar dos registros para cada número: un registro con los 8 bits más significativos y otro con los 8 bits menos significativos. Justamente, queremos extender el computador básico para permitir esto.

Estos números, con el doble del número de bits que los números "normales", suelen llamarse `long` en los lenguajes de programación. Por lo tanto, los nombres de todas las instrucciones que agreguemos a nuestro assembly para manejar estos números van a empezar con la letra "L" mayúscula; p.ej., LADD (para sumar), LSHL (para hacer shift a la izquierda).

Los registros A y B actuales van a seguir tal cual, excepto que cuando manejemos números `long`, ellos van a almacenar los bits menos significativos de los números involucrados. Los bits más significativos van a ser almacenados en dos registros adicionales, LA y LB. Por lo tanto, para poner un valor `long` en los registros, primero hay que cargar el registro A (o B) y luego LA (o LB).

Al sumar dos números `long`, podríamos sumar primero los registros A y B, y luego los registros LA y LB, pero probablemente esto sería muy lento. Por esto, vamos a incorporar una segunda ALU, llamémosla LALU, para ejecutar la suma de LA y LB "simultáneamente" con la suma de A y B.

(a) Dibuja y explica de manera clara y precisa los cambios necesarios en el circuito del computador básico: destaca claramente componentes, cables, buses de datos nuevos/modificados y cualquier otro cambio que hagas. La ALU no cambia en cuanto a que sigue teniendo dos inputs, A y B, y un output, Result, y que todos son de 8 bits. Sin embargo, es posible que tenga que interactuar con la LALU. **(3 pts)**

(b) Especifica las nuevas instrucciones que permitan sumar dos números `long` —según las especificaciones anteriores— en cuanto a sus nombres, las señales de control necesarias, y la operación que realizan, similarmente a las otras instrucciones que ya conocemos y de manera coherente con tu dibujo en (a). **(3 pts)**

4 Assembly del computador básico

Responda las siguientes preguntas para el assemby del computador visto en clases, para el *script* que se presenta a continuación de estas. Las direcciones de memoria van desde $0x00$ a la $0xFF$

(a) Que valores están almacenado en los registros, stack y memoria de datos en la primera llamada a `sub2` justo antes de la línea 25. **(2 pts)**

(b) Que valores están almacenado en los registros, stack y memoria de datos en la primera llamada a `sub1` justo antes de la línea 43. **(2 pts)**

(c) Que valores están almacenado en los registros, stack y memoria de datos en la segunda llamada a `sub1` justo antes de la línea 43. **(2 pts)**

```
.data
    a 3
    r 0
    i 0
.text
    MOV A, (a)
    MOV B, (r)
    CMP A,B
    JEQ end1
    ADD A, (a)
    PUSH A
    PUSH B
    CALL sub1
    POP A
    POP B
    MOV A, 0
    CMP A, 0
    JEQ end2
    sub2:
        ADD B, A
        SUB A, 1
        CMP A, 0
        BGT sub2
        MOV B, (r)
        RET
    sub1:
        MOV A, (r)
        MOV B, (r)
        INC B
        PUSH A
        PUSH B
        CALL sub2
        POP B
        POP A
        MOV A, (i)
        INC A
        MOV (i), A
        MOV B, (a)
        CMP A, B
        BEQ end_sub1
        PUSH A
        PUSH B
        CALL sub1
        POP B
        POP A
        end_sub1:
            RET
    end1:
        ADD A, 1
        MOV (r), A
    end2:
```

Las siguientes tablas tienen algunas potencias de 2 que podrían ser útiles para el desarrollo de la pregunta 1. Los casilleros pueden ser utilizados para responder las preguntas 1.2 y 1.3 pero no es obligatorio.

2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}
0.5	0.25	0.125	0.0625	0.03125	0.015625	0.0078125	0.00390625

2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
32768	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1

2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}
8388608	4194304	2097152	1048576	524288	262144	131072	65536

2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}
2147483648	1073741824	536870912	268435456	134217728	67108864	33554432	16777216

$n_1 =$

$n_2 =$

$n_1 + n_2 =$

$-5.451 \times 10^4 =$