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

IIC2343 – Arquitectura de Computadores (I/2013)

Interrogación 2

Pregunta 1

a) Al iniciar el cuerpo de una subrutina, ¿por qué es necesario ejecutar las instrucciones PUSH BP y MOV BP, SP? ¿Qué pasa si no se ejecutan? **(1 pto.)**
Solución: La primera instrucción respalda el valor previamente almacenado en BP, de manera de poder recuparlo si existen llamados a subrutinas anidadas. La segunda instrucción crea un nuevo valor para BP, basado en el actual valor de SP, que indica el tope del stack. Esto permite usar al registro BP como referencia para el direccionamiento de parámetros y variables locales. En caso de no ejecutarse estas instrucciones, no es posible usar al registo BP como referencia, además de no tener seguridad de poder recuperar el valor previo de BP.

b) ¿Qué instrucciones podrían agregarse al computador básico si se agrega al MUX A una entrada para el bus de literales? **(1 pto.)**
Solución: Se podrían agregar instrucciones de transferencia y direccionamiento que involucren al registro B y a un literal de manera simultánea. Esto no era posible antes, ya que el bus de literales y el registro B estaban conectados al mismo MUX.

c) ¿Cómo se puede evitar la sobreescritura accidental de la memoria de datos por el uso del stack en el computador básico? **(1 pto.)**
Solución: Una de las posibles soluciones, es agregar un límite arbitrário al tamaño del stack y al tamaño total usado por las variables. De estas manera, si los tamaño están bien puestos, el tamaño del stack no podrá crecer lo suficiente como para sobreescriba variables.

d) ¿Es posible realizar cálculos con números de punto flotante en el computador básico? **(1 pto.)**
Solución: De manera nativa no es posible hacer cálculos de punto flotante en el computador, ya que no tiene FPU. Para hacer entonces, es necesario emular números de punto flotante mediante software.

e) Enumere y describa las reglas de la convención de llamada *stdcall*. **(1 pto.)**
Solución:

   a) Parámetros son introducidos en el stack de derecha a izquierda.
   b) El registro SP debe quedar apuntando a la misma posición que estaba antes de introducir los parámetros.
   c) El valor de retorno se almacena en el registro AX.

f) Describa situaciones en que se prefieran ISAs RISC sobre CISC y viceversa. **(1 pto.)**
Solución: Una ISA RISC es preferida en situaciones en que el consumo de energía es un aspecto crítico, por lo que se necesita hardware muy simple y de poco consumo. De manera análoga, en situaciones en que lo principal es el rendimiento y el aspecto energético no es el central, se prefiere una ISA CISC.

Pregunta 2

En las siguientes preguntas, cualquier detalle de implementación o aspecto que se asuma debe quedar claramente explicado.

a) Escriba en el assembly del computador básico un programa que calcule el factorial de un número natural n. **(3 ptos.)**

Solución:

```
DATA
n         ?
index     2
i         0
fact      1
mul_res   0

CODE
MOV A, (n)
CMP A, 1
JLE end

while:
MOV A, (n)
MOV B, (index)
CMP A, B
JLT end

MOV A, 0
MOV (i), A
MOV (mul_res), A
mul:
MOV A, mul_res
ADD A, (fact)
MOV (mul_res), A
MOV A, (i)
ADD A, 1
MOV (i), A
MOV B, (index)
CMP A, B
JLT  mul

ADD B, 1
MOV (index), B
MOV A, (mul_res)
MOV (fact), A
JMP while

end:
```

b) Escriba en assembly x86, usando al menos una subrutina, la convención *stdcall* y soporte para variables locales (reg. BP), un programa que realice la búsqueda de un número en un arreglo ordenado, utilizando el algoritmo de búsqueda binaria. Para implementar este algoritmo, se compara el elemento a buscar con el elemento central del arreglo. Si el valor de este elemento es mayor que el del buscado, se repite el procedimiento en la parte del arreglo que va desde el inicio de éste hasta el elemento central. En caso contrario, se toma la parte del arreglo que va desde el elemento central hasta el final. **(3 ptos.)**

Solución: Se asume que el arreglo tendrá un tamaño par y mayor que cero y que el número buscado siempre estará en el arreglo.

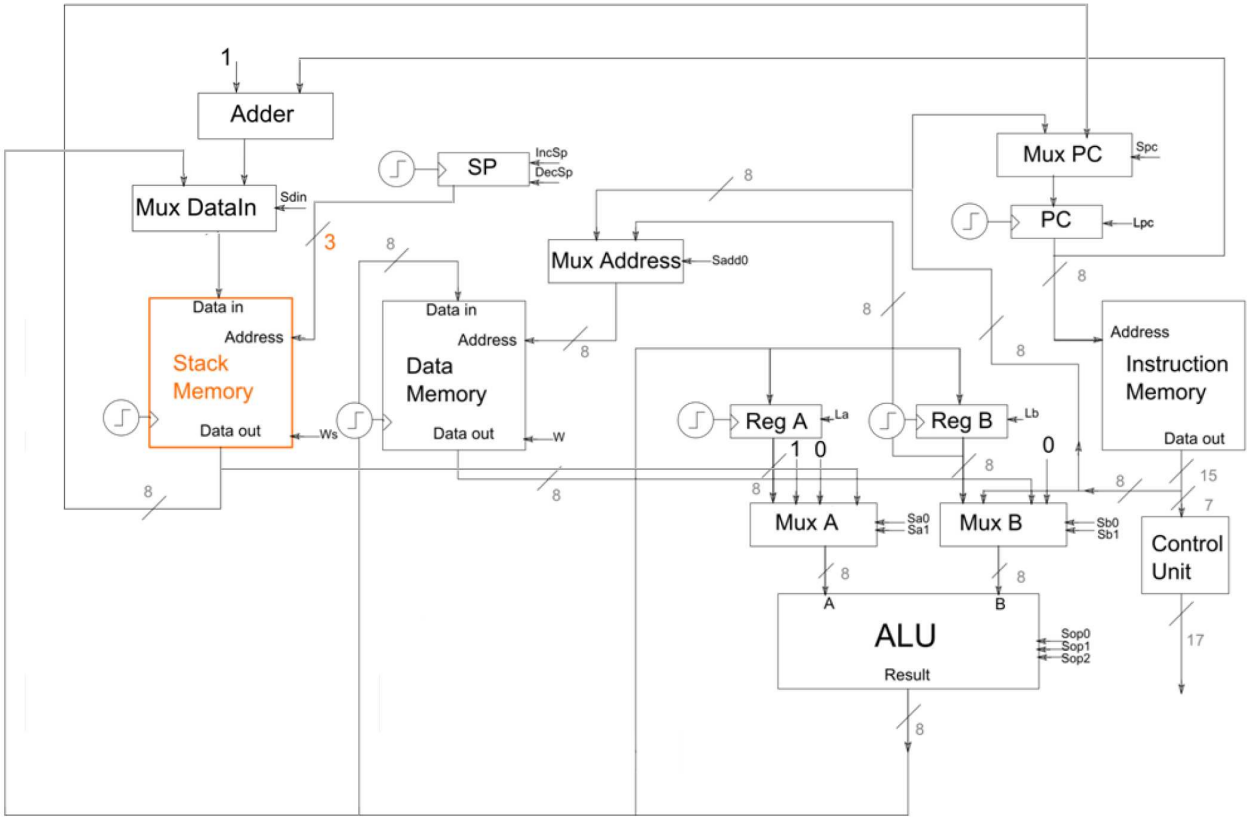
```
        MOV AX, 0
        MOV AL, target
        PUSH AX
        MOV AL, n
        PUSH AX
        LEA AX, array
        PUSH AX
        CALL binary_search
        MOV res, AX
        RET

        binary_search:
        PUSH BP
        MOV BP, SP
        MOV BX, [BP + 4]
        MOV SI, [BP + 6]
        MOV CX, [BP + 8]

        start:
        SHR SI, 1
        CMP [BX + SI], CL
        JE end
        JG start
        ADD BX, SI
        JMP start

        end:
        MOV AX, BX
        ADD AX, SI
        POP BP
        RET 6

        res    dw
        target db    ?
        n      db    ?
        array  db    ?
```

Pregunta 3

Se desea modificar el computador básico a nivel de microarquitectura e ISA. Para los siguientes puntos detalle las modificaciones que haría. Utilice diagramas de componentes y conexiones y tablas de opcodes e instrucciones cuando corresponda. En caso que sus soluciones utilicen más de un ciclo de clock por instrucción, los valores previamente almacenados en los registros no deben perderse.

a) Tener stack de uso general e independiente de la memoria de datos. **(2 ptos.)**

Solución: A continuación se muestran las modificaciones realizadas a la microarquitectura del computador básico.

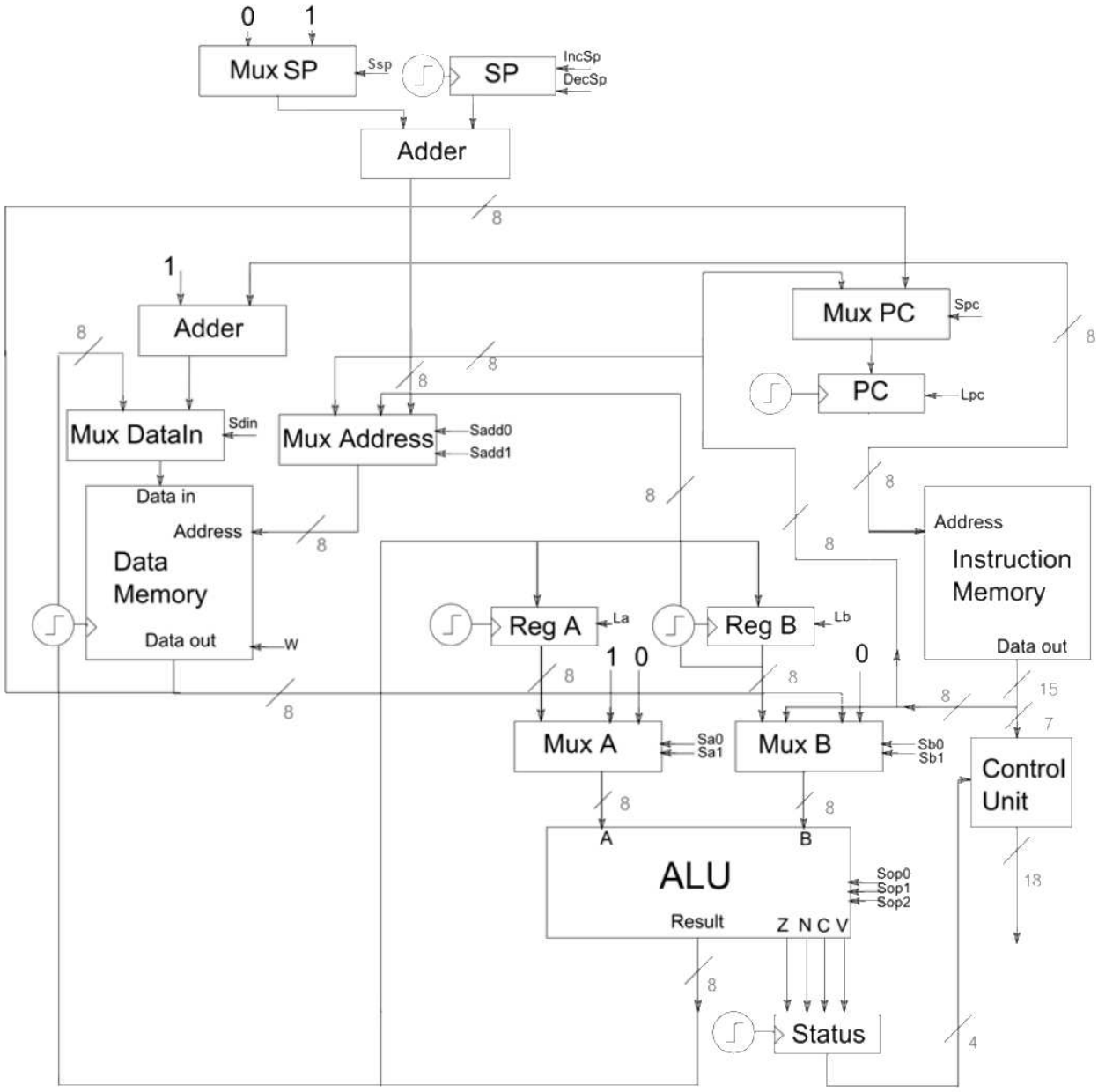


En el diagrama puede apreciarse la eliminación de las señal de control Sdin, que ya no es necesaria, ya que la dirección en el stack vendrá siempre desde el registro SP. Además, se agrega la señal Ws, que controla la escritura en el stack. Para poder soportar la extracción de datos del stack para almacenarlos en los registros, se extiende la linea de salida del stack hasta el MUX A. A nivel de ISA, los cambios necesarios son unicamente la modificación de la señales de control correspodientes a PUSH, POP, RET y CALL. Estas instrucciones deben ahora tomar en cuenta el uso de la señal Ws y la nueva entrada del MUX A, como se muestra a continuación:

Inst.	Op.	Opcode	Lpc	La	Lb	Sa0,1	Sb0,1	Sop0,1,2	Sadd0	Spc	W	Ws	Sdin	IncSp	DecSp
CALL	Dir	1000101	1	0	0	-	-	-	-	LIT	0	1	PC	0	1
RET		1000110	0	0	0	-	-	-	-	-	0	0	-	1	0
		1000111	1	0	0	-	-	-	-	DOUT	0	0	-	0	0
PUSH	A	1001000	0	0	0	A	0	ADD	-	-	0	1	ALU	0	1
PUSH	B	1001001	0	0	0	0	B	ADD	-	-	0	1	ALU	0	1
POP	A	1001010	0	1	0	-	-	-	-	-	0	0	-	1	0
		1001011	0	1	0	DOUT	0	ADD	-	-	0	0	ALU	0	0
POP	B	1001100	0	0	0	-	-	-	-	-	0	0	-	1	0
		1001101	0	0	1	DOUT	0	ADD	-	-	0	0	ALU	0	0

b) Permitir al computador básico la ejecución de todas sus instrucciones en un sólo ciclo del clock. **(2 ptos.)**

Solución: Para permitir la ejecución en un solo ciclo del clock, es necesario asegurar que POP y RET se ejecuten en un ciclo. Para lograrlo, se agrega un Adder, el cual recibe un 1 o un 0 desde el nuevo MuxSP, que tiene una entrada, puesta en 1 y otra en 0, dependiendo si la instrucción ejecutada es alguna de estas últimas dos. De esta manera, se obtiene de manera inmediata el valor correcto desde la memoria y no hay que esperar un ciclo más para hacerlo. A continuación se muestran las modificaciones realizadas a la microarquitectura:

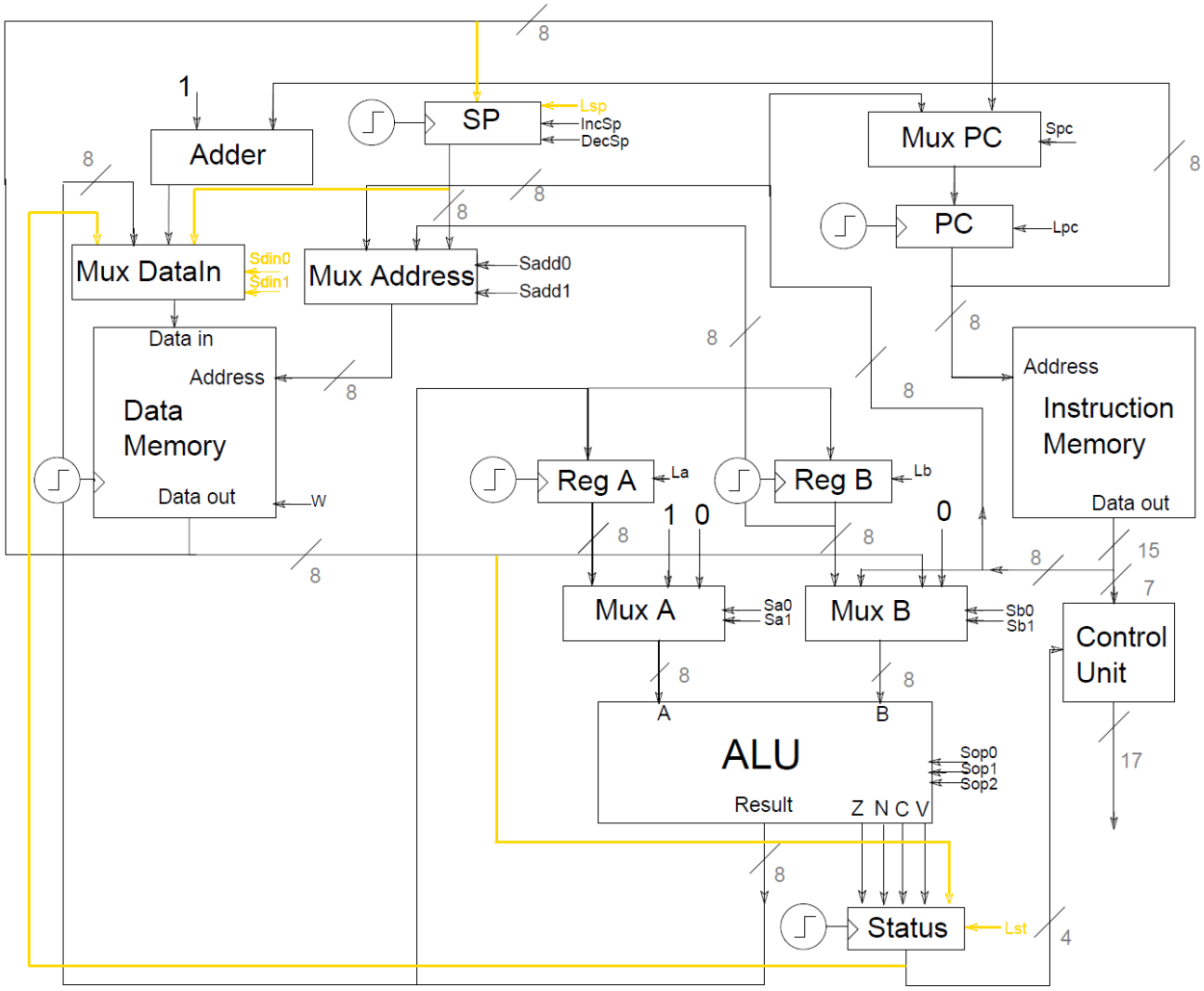


Desde el punto de vista de la ISA, es necesario modificar las señales de control de esas instrucciones, para que activen ahora la señal Ssp cuando se utilicen alguna de estas dos instrucciones. Esto significa que ahora las instrucciones POP y RET setearán la señal Ssp en 1, mientras que el resto de las instrucciones lo seteará en 0.

Inst.	Op.	Opcode	Lpc	La	Lb	Sa0,1	Sb0,1	Sop0,1,2	Sadd0,1	Sdin0	Spc0	W	IncSp	DecSp	Ssp
RET		1000110	1	0	0	-	-	-	SP	-	DOUT	0	1	0	1
POP	A	1001010	0	1	0	0	DOUT	ADD	SP	-	-	0	1	0	1
POP	B	1001100	0	0	1	0	DOUT	ADD	SP	-	-	0	1	0	1

c) Soportar las instrucciones PUSHALL y POPALL. PUSHALL almacena en el stack el contenido de todos los registros (A,B,SP,PC,Status), mientras que POPALL rescata desde el stack estos valores y los vuelve a poner en los registros correspondientes. **(2 ptos.)**

Solución: Estas instrucciones pueden soportarse facilmente utilizando 5 opcodes para cada una, donde cada opcode se relaciona con el respaldo/carga de uno de los registros. Para soportar esto desde el punto de vista del hardware, es necesario crear conexiones desde los registros SP y Status, para que sus valores sean respaldados/cargados.



Para la ISA, debemos utilizar los opcodes creados anteriormente para respaldo/carga de los registros A, B y PC, y además se necesitan 4 opcodes nuevos para respaldo/carga de los registros SP y Status.

Inst.	Op.	Opcode	Lpc	La	Lb	Sa0,1	Sb0,1	Sop0,1,2	Sadd0,1	Sdin0,1	Spc0	W	IncSp	DecSp	Lsp	Lst
PUSHALL		1001110	0	0	0	-	-	-	SP	PC	-	1	0	1	0	0
		1001000	0	0	0	A	0	ADD	SP	ALU	-	1	0	1	0	0
		1001001	0	0	0	0	B	ADD	SP	ALU	-	1	0	1	0	0
		1001111	0	0	0	-	-	-	SP	ST	-	1	0	1	0	0
		1010000	0	0	0	-	-	-	SP	SP	-	1	0	1	0	0
POPALL		1001100	0	0	0	-	-	-	-	-	-	0	1	0	0	0
		1010001	0	0	0	0	-	-	SP	-	-	0	0	0	1	0
		1001100	0	0	0	-	-	-	-	-	-	0	1	0	0	0
		1010010	0	0	0	0	-	-	SP	-	-	0	0	0	0	1
		1001100	0	0	0	-	-	-	-	-	-	0	1	0	0	0
		1001101	0	0	1	0	DOUT	ADD	SP	-	-	0	0	0	0	0
		1001010	0	0	0	-	-	-	-	-	-	0	1	0	0	0
		1001011	0	1	0	0	DOUT	ADD	SP	-	-	0	0	0	0	0
		1000110	0	0	0	-	-	-	-	-	-	0	1	0	0	0
		1000111	1	0	0	-	-	-	SP	-	DOUT	0	0	0	0	0