Complemento de Arquitectura de Computadoras

Examen Diciembre 2016

- **Indique su nombre completo y número de cédula en cada hoja.**
- **Numere todas las hojas e indique el total de hojas en la primera. Escriba las hojas de un solo lado.**
- **Solo se responderán dudas de letra. No se responderán preguntas en los últimos 30 minutos de la prueba.**
- **La prueba es individual y sin material. Duración de la prueba : 3 horas.**
- **El puntaje mínimo de aprobación es de 60 puntos.**
- **Justifique todas sus respuestas.**

Pregunta 1 (10 puntos)

En un sistema con caché, describa y compare los algoritmos de escritura a memoria write-through y write-back. Analice cómo impactan en el problema de coherencia de caché.

Pregunta 2 (10 puntos)

En un sistema con paginación, explique cómo se gestiona el fallo de página.

Pregunta 3 (10 puntos)

Defina y compare las métricas “Tiempo de CPU” y “MIPS” para medir rendimiento.

Pregunta 4 (10 puntos)

Proponga un ejemplo donde es más eficiente procesar la E/S con DMA que con interrupciones.

Problema 1 (30 puntos)

Considere la siguiente estructura de datos para manejar inscripciones a cursos en un sistema x86:

```
struct _asignatura{
        short codA;
        char* nombre;
}
struct curso{
        short año;
        _asignatura* asig;
        short cantEstudiantes;
}
```

Y las siguientes variables globales:

```
short cantCursos;
curso[1024] cursos;
```

Donde la variable 'cantCursos' indica la cantidad de posiciones válidas del arreglo.

a. Implementar en un lenguaje de alto nivel, la función:

   short cantidadTotalEstudiantes(short año);

   Que dado un año, indica la cantidad total de estudiantes en cursos de dicho año.

Solución:

```
short cantidadTotalEstudiantes( short año){
      short cant = 0;
      for (int i = 0; i < cantCursos; i++){
            if (cursos[i].año == año){
```

```
                cant += cursos[i].cantEstudiantes;
            }
        }
        return cant;
    }
```

b. Representar gráficamente un elemento del struct 'curso' en memoria y calcular la cantidad de bytes necesaria para almacenar el arreglo de cursos. Suponga que los punteros se guardan como desplazamientos con respecto al segmento ES.

Solución:

Muestro elemento de struct 'curso' a partir de dirección 0x2000.

0x2000	curso.año (parte baja)
0x2001	curso.año (parte alta)
0x2002	curso.asig (parte baja)
0x2003	curso.asig (parte alta)
0x2004	curso.cantEstudiantes (parte baja)
0x2005	curso.cantEstudiantes (parte alta)

Por cada elemento del arreglo se precisan 6 bytes, por lo tanto para el arreglo entero se precisan 1024 * 6 = 6KB.

c. Compilar en assembler 8086 sabiendo que las variables globales se encuentran almacenadas a partir de la dirección 0 del ES. La variable año se recibe en el stack y el resultado se devuelve en el stack. Se debe preservar el valor de todos los registros.

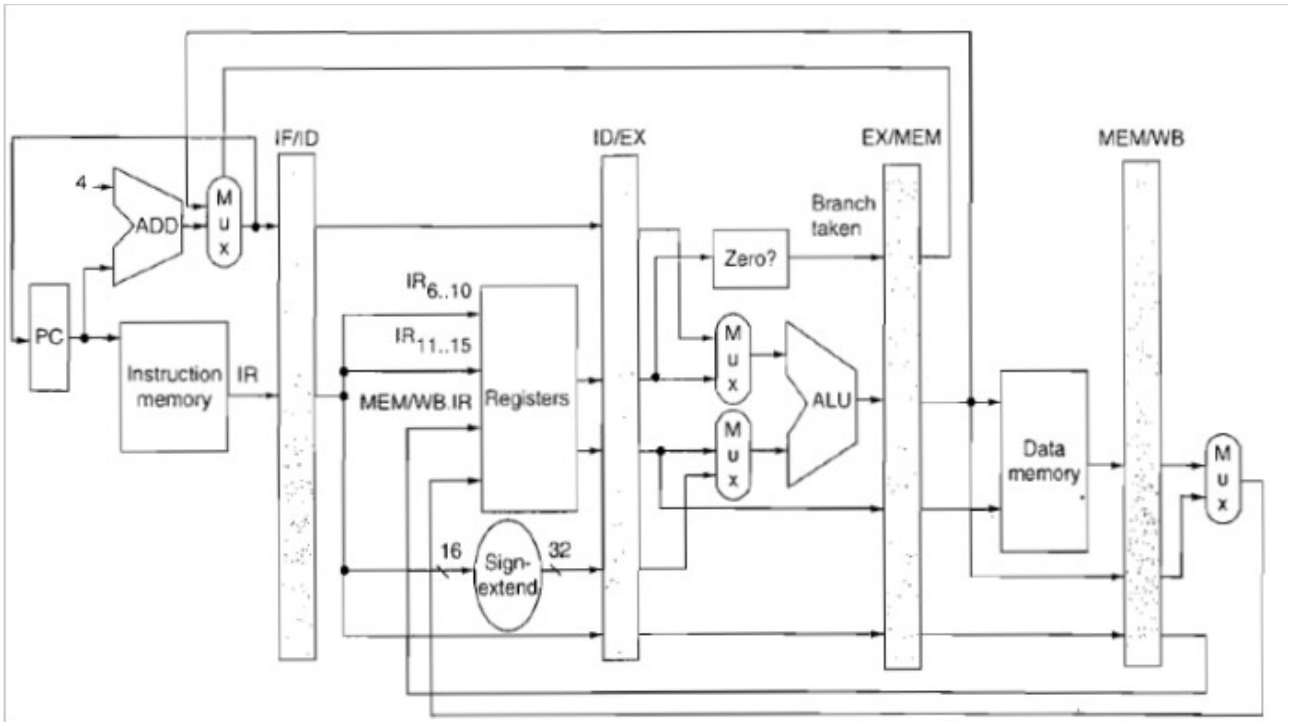
Solución:

```
cantidadTotalEstudiantes PROC
    PUSH BP
    MOV BP, SP
    PUSH AX
    PUSH BX
    PUSH CX
    PUSH DX
    XOR AX, AX          ; cant = 0
    XOR DX, DX          ; i = 0
    MOV BX, 2           ; puntero a memoria
    MOV CX, [BP + 4]    ; cx = año
FOR:
    CMP DX, ES:[0]
    JL   FINFOR
    CMP CX, ES:[BX]
    JNE  FINIF
    ; mismo año, tengo que sumar
```

```
            ADD AX, ES:[BX + 4]
FINIF:
            ADD BX, 6       ; apunto al siguiente elemento de cursos
            INC DX
            JMP FOR
FINFOR:
            MOV [BP + 4], AX        ; guardo resultado
            POP DX
            POP CX
            POP BX
            POP AX
            POP BP
            RET
```

Problema 2 (30 puntos)

Considere la implementación del Pipeline MIPS visto en el curso:



Suponga que en este pipeline se ejecuta el siguiente grupo de instrucciones:

```
loop:
     LD   R1, 0(R2) ; carga en R1 el valor de la dirección 0 + R2
     ADDI R1, R1, #1 ; R1 = R1 + 4
     SD   R1, 0 (R2) ; guarda el valor de R1 en la dirección 0+R2
```

```
ADDI R2, R2, #8 ; R2 = R2 + 8
SUBI R4, R3, #8 ; R4 = R3 - R2
BNEZ R4, loop ; saltar a loop si R4 != 0
```

a. Realice un diagrama del Pipeline para la ejecución de una primera vuelta del loop (desde que inicia la instrucción LD hasta que finaliza la instrucción BNEZ). **Nota**: como se muestra en el dibujo, no hay forwarding implementado.

b. Repita el diagrama anterior, suponiendo una implementación completa de forwarding.

c. Calcule la aceleración lograda con la implementación de forwarding para la porción de código indicada en las partes anteriores.

d. A partir del diagrama del pipeline realizado en la parte b), extiéndalo para completar dos iteraciones del loop.

e. Calcule los ciclos necesarios para ejecutar 100 iteraciones del loop.

Solución:

a.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
LD R1, 0(R2)	IF	ID	EX	MEM	WB											
ADDI R1, R1, #1		IF	ID	ID	ID	EX	MEM	WB								
SD R1, 0(R2)			IF	IF	IF	ID	ID	ID	EX	MEM	WB					
ADDI R2, R2, #8						IF	IF	IF	ID	EX	MEM	WB				
SUBI R4, R3, #8									IF	ID	EX	MEM	WB			
BNEZ R4, loop										IF	ID	ID	ID	EX	MEM	WB

b.

	1	2	3	4	5	6	7	8	9	10	11
LD R1, 0(R2)	IF	ID	EX	MEM	WB						
ADDI R1, R1, #1		IF	ID	ID	EX	MEM	WB				
SD R1, 0(R2)			IF	IF	ID	EX	MEM	WB			
ADDI R2, R2, #8					IF	ID	EX	MEM	WB		
SUBI R4, R3, #8						IF	ID	EX	MEM	WB	
BNEZ R4, loop							IF	ID	EX	MEM	WB

c.

speedup = t_viejo / t_nuevo

Como el ciclo de reloj y la cantidad de instrucciones no cambia, equivale a dividir los ciclos de reloj totales.

Speedup = 16 / 11 = 1.45

d.

	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
(1er loop)															
ADDI R1, R1, #1	WB														
SD R1, 0(R2)	MEM	WB													
ADDI R2, R2, #8	EX	MEM	WB												
SUBI R4, R3, #8	ID	EX	MEM	WB											
BNEZ R4, loop	IF	ID	EX	MEM	WB										
(2ndo loop)															
LD R1, 0(R2)		-	-	-	IF	ID	EX	MEM	WB						
ADDI R1, R1, #1						IF	ID	ID	EX	MEM	WB				
SD R1, 0(R2)							IF	IF	ID	EX	MEM	WB			
ADDI R2, R2, #8									IF	ID	EX	MEM	WB		
SUBI R4, R3, #8										IF	ID	EX	MEM	WB	
BNEZ R4, loop											IF	ID	EX	MEM	WB

e.

Por lo calculado en la parte b), la primer vuelta del loop dura 11 ciclos de reloj. Por lo calculado en al parte d), cada vuelta adicional dura 10 ciclos de reloj. Por lo tanto, para ejecutar 100 iteraciones, se precisan:

ciclos_totales = 11 + 99 * 10 = 1001 ciclos de reloj.