EIMT.UOC.EDU

PEC1

Estructura de computadores

Programa

2021 s2

Estudios de informática, multimedia y comunicación

EIMT.UOC.EDU

Presentación

La presente PEC1 contiene 5 preguntas y representa el 50% de la nota de la evaluación contínua.

Como podréis ver, los ejercicios son muy parecidos a los que habéis hecho durante estos dias, en los que además habéis podido dar las soluciones, comentarlas y plantear dudas en el foro. Esta PEC es **individual**, **evaluable** y por tanto no puede comentarse.

Competencias

Las competencias específicas que persigue la PEC1 son:

- [13] Capacidad para identificar los elementos de la estructura y los principios de funcionamiento de un ordenador.
- [14] Capacidad para analizar la arquitectura y organización de los sistemas y aplicaciones informáticos en red.
- [15] Conocer las tecnologías de comunicaciones actuales y emergentes y saberlas aplicar convenientemente para diseñar y desarrollar soluciones basadas en sistemas y tecnologías de la información.

Objetivos

Los objetivos de la siguiente PEC son:

- Conocer el juego de instrucciones de la máquina CISCA.
- Conocer los modos de direccionamiento de la máquina CISCA.
- Traducir pequeños programas a ensamblador.
- Comprender la codificación interna de las instrucciones de ensamblador.
- Describir las micro operaciones involucradas en la ejecución de una instrucción de ensamblador.

Enunciado

Responder cada pregunta o apartado en el recuadro correspondiente.

Recursos

Podéis consultar los recursos disponibles en el aula, pero no hacer uso del foro.

Criterios de valoración

La **puntuación** de cada pregunta y los **criterios de evaluación** los encontraréis a cada pregunta.

Formato y fecha de entrega

- La PEC1 debéis entregarla en el apartado de **entrega de actividades** con el nombre **apellido1_apellido2_nombre_PEC1 (pdf / odt / doc / docx).**
- La fecha **límite** de entrega es el **14/03/2022.**

Enunciado

Pregunta 1 (2 puntos)

Suponed que el estado inicial del computador (el valor que contienen los registros, posiciones de memoria y bits de resultado justo antes de comenzar la ejecución de cada fragmento de código, de cada apartado) es el siguiente:

R0= 000h R1= 100h R2= 200h R3= 003h R4= 004h	M(00000100h)=00001111h M(00000200h)=FFFF0000h M(00000003h)=11110000h M(00000004h)=FFFFF000h M(00000005h)=F000000Fh M(00000A00h)=AAAAAAAAh

- Bits de resultado del registro de estado: Z=0, S=0, C=0, V=0
- Registros especiales: suponemos que el PC apunta al inicio del fragmento de código de cada apartado.

¿Cuál será el estado del computador después de ejecutar cada uno de los siguientes fragmentos de código? Indicad solamente el contenido (**en hexadecimal**) de los registros y posiciones de memoria que se hayan modificado como resultado de la ejecución del código. Indicad el valor final de todos los bits de resultado. (No os pedimos que indiquéis el valor del PC después de ejecutar el código y por eso no os hemos dado el valor inicial del PC, donde comienza cada fragmento de código).

Suponed que la dirección simbólica A vale 00000A00h

Importante: dejad claramente indicado (**preferiblemente resaltado o en otro color**) **al final de cada apartado el valor final** de los registros y direcciones de memoria modificadas, así como los bits de estado.

a)

```
MOV  R0,[R2]
ADD  [A], R0
MOV  R0, [A]
```

R0:= [200H]= FFFF0000H
[A]=[A00H]= AAAAAAAAh + FFFF0000H = AAA9AAAAH
R0:= AAA9AAAAH

Z= 0 , S= 1 , C= 1 , V= 0

b)

```
ADD R1, R2
SUB R1,[R3]
NOT R1
```

R1:= 100h + 200h = 300h
R1:= 300h – 11110000h = EEEF0300h
R1:= 1110FCFFh

Z= 0 , S= 1 , C= 1 , V= 0

c)

```
      MOV   R1, 0
CONT: CMP   R3, 6h
      JE    END
      ADD   R1, [R3]
      ADD   R3, 1
      JMP CONT
END:
```

```
R1:= 0
; R3= 3
R1:= 0 + 11110000h = 11110000h
R3:= 4

; R3= 4
R1:= 11110000h + FFFFF000h = 1110F000h
R3:= 5

; R3= 5
R1:= 1110F000h + F000000Fh = 0110F00Fh
R3:= 6

; R3= 6

Z=  1  , S=  0  , C=  0  , V=  0
```

Criterios de valoración. Los apartados a) y b) valen 0,5 puntos cada uno y el c) vale 1 punto. La valoración de cada apartado es del 100% si no hay ningún error en la solución del apartado, es del 50% si el contenido de las posiciones de memoria y registros modificados es correcto, pero hay algún error en el contenido de uno o varios de los bits de resultado, y es del 0% si hay algún error en alguna posición de memoria o registro modificado.

Pregunta 2 (2 puntos)

En el código de alto nivel `M` es una variable de tipo vector de 10 elementos. Cada elemento de la matriz es un entero de 32 bits. En el programa ensamblador la matriz se encuentra almacenada a partir de la dirección simbólica `M`, en posiciones consecutivas (`M[0]`, en la dirección simbólica `M`, el siguiente, `M[1]`, en la dirección `M+4`, etc.).

SOURCE es una variable de tipo vector de una dimensión de 10 elementos. Cada elemento del vector es un entero de 32 bits. En el programa ensamblador la matriz se encuentra almacenada a partir de la dirección simbólica `SOURCE` en posiciones consecutivas (SOURCE`[0]`, en la dirección simbólica `SOURCE`, el siguiente, SOURCE`[1]`, en la dirección SOURCE`+4`, etc.).

El programa recorre los 2 vectores actulizando M si es mayor que 0 con los valores de SOURCE mientras no encontremos un 0 en SOURCE.

```
I= 9;
SOURCE[0]= 0;
DO {
     IF (M[I]>0) M[I]= SOURCE[I];
     I:= I-1;
} WHILE (SOURCE[I]!=0)
```

Rellena los espacios de la propuesta de programa ensamblador que se muestra a continuación para conseguir el resultado deseado.

```
Solución:

          MOV R1, 36
          MOV [SOURCE], 0
          MOV R3,  M
          ADD R3, R1
DO:       CMP [R3], 0
          JLE WHI
          MOV R4,[SOURCE+R1]
          MOV [R3] , R4
WHI:      SUB R3, 4
          SUB R1, 4
          CMP [SOURCE+R1], 0
          JNE DO
```

Criterios de valoración.

2 puntos. Se pierden 0,5 puntos por cada instrucción incorrecta.

Pregunta 3 (3 puntos)

Traducid a lenguaje máquina el fragmento de código en lenguaje ensamblador que os proponemos en la tabla.

Suponed que la primera instrucción del código se ensambla a partir de la dirección **0033CC00h** (que es el valor del PC antes de empezar la ejecución del fragmento de código). Suponed que la dirección simbólica V vale **00040404h**

			Bk por k=0..10										
Dirección	**Eti**	**Ensamblador**	0	1	2	3	4	5	6	7	8	9	10
0033CC00h	E1:	SAL R1, 1h	35	11	00	01	00	00	00				
0033CC07h		CMP [R2+A0h].R1	26	42	A0	00	11						
0033CC0Ch		JE E2	41	60	0D	00							
0033CC10h		ADD R1, [V]	20	11	20	04	04	04	00				
0033CC17h		JMP E1	40	00	00	CC	33	00					
0033CC1Dh	E2:												

Criterios de valoración. Cada instrucción ensamblada incorrectamente resta 0.5 puntos. Una instrucción está incorrectamente ensamblada si no se escribe el valor o se escribe un valor incorrecto de uno o varios de los dígitos hexadecimal que codifican la instrucción en lenguaje máquina. Además, se resta 0.5 puntos si hay algún error en la columna que indica las direcciones de memoria en las que comienza cada instrucción.

Pregunta 4 (2 puntos)

El *ciclo de ejecución* de una instrucción se divide en 3 fases principales
1) Lectura de la instrucción
2) Lectura de los operandos fuente
3) Ejecución de la instrucción y almacenamiento del operando destino

Dar la secuencia de micro-operaciones que hay que ejecutar en cada fase para las siguientes instrucciones del código codificado en la pregunta anterior.

ADD R1, [V]

Fase	Micro-operaciones
1	MAR:= 0033CC10h, READ; Ponemos el contenido de PC en el registro MAR. MBR:= 00040404201120; Leemos la instrucción PC:= 0033CC17h; Incrementamos el PC en 7 unidades (tamaño de la instrucción). IR:= MBR; Cargamos la instrucción en el registro IR.
2	MAR:= Contenido IR(V) , READ MBR:= memoria
3	R1:= R1 + MBR

JMP E1

Fase	Micro-operaciones
1	MAR:= 0033CC17h, READ; Ponemos el contenido del PC en MAR MBR:= 0033CC000040; leemos la instrucción PC:= 0033CC1Dh; incrementamos el PC en 6 unidades IR:= MBR; Cargamos la instrucción en IR
2	No necesitamos operaciones para obtener el operando.
3	PC= dirección de salto

Criterios de valoración. Si todo está correcto se obtienen 2 puntos. Cada instrucción es independiente y vale 1 punto. Se resta 0.5 por cada fallo dentro de la misma instrucción.

Pregunta 5 (1 punto)

Responde a las siguientes preguntas:

Pregunta 1. ¿Qué es el borrow y cómo se calcula?

El «borrow» es en la resta el equivalente al acarreo en la suma: nos llevamos una. Dado que la resta se implementa como suma del sustrayendo en Ca2, el bit de transporte de la resta (borrow) será el bit de transporte (carry) negado obtenido de hacer la suma con el complementario.

Pregunta 5.2. ¿Qué diferencia hay entre las instrucciones de salto condicional y las de salto incondicional?

Las instrucciones de salto condicional rompen la secuencia saltando a una dirección destino resultado de sumar un desplazamiento a la dirección actual del PC. Además, el salto se realiza evaluando previamente una condición basada en los bits de estado.
Las instrucciones de salto incondicional, saltan a una dirección determinada directamente si evaluar ninguna condición.
Las instrucciones de salto incondicional permiten saltos más largos que las de salto condicional.

Criterios de valoración. Cada pregunta individual vale 0,5 puntos si está correcta (respuesta razonada y que conteste a lo que se pregunta). 0 si no está correcta o es incompleta.