Guías de Sistemas de Computación II

2020

Se incluyen a continuación:

Trabajos Prácticos

Los trabajos prácticos se realizan en forma individual por los alumnos. Están diseñados en base a contenidos cubiertos por la bibliografía obligatoria. Permiten ejercitar en forma sistemática, los distintos puntos de las unidades del programa.

Los Trabajos Prácticos 1, 2 y 3 se centran en los contenidos de las Unidades 1, 2 y 3. El Trabajo Práctico 4 se centra en los contenidos de las Unidades 4, 5 y 6.

Preguntas para abordar el aprendizaje

Las preguntas para abordar el aprendizaje están orientadas a que el alumno enfoque el estudio de las unidades con la misma extensión y profundidad que se requiere para aprobar la materia.

El examen final podrá incluir cualquier tema del programa, y tendrá una orientación y detalle similar al de las preguntas que aquí se presentan.

Se recomienda al alumno, no presentarse a dar el examen final si no puede responder la totalidad de las preguntas aquí desarrolladas.

Abordaje bibliográfico

En la planilla que se incluye se detallan la totalidad de los puntos descriptos en cada una de las unidades del programa, asociados a las distintas obras bibliográfica en donde los mismos pueden profundizarse. La bibliografía se referencia mediante Capítulos o Títulos dentro de cada capítulo y la letra “p” al lado del número de capítulo o título indica que la bibliografía trata el punto del programa en forma “parcial”

Mapa conceptual

Se incluye en un mapa, la relación de los distintos temas tratados en la materia.

Guía de Trabajos Prácticos 2020

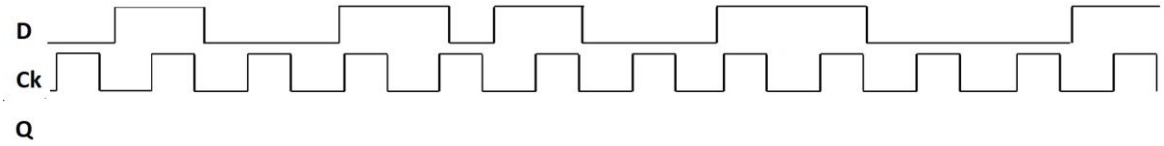
UNIVERSIDAD ABIERTA INTERAMERICANA
FACULTAD DE TECNOLOGÍA INFORMÁTICA

Asignatura: Sistemas de Computación II

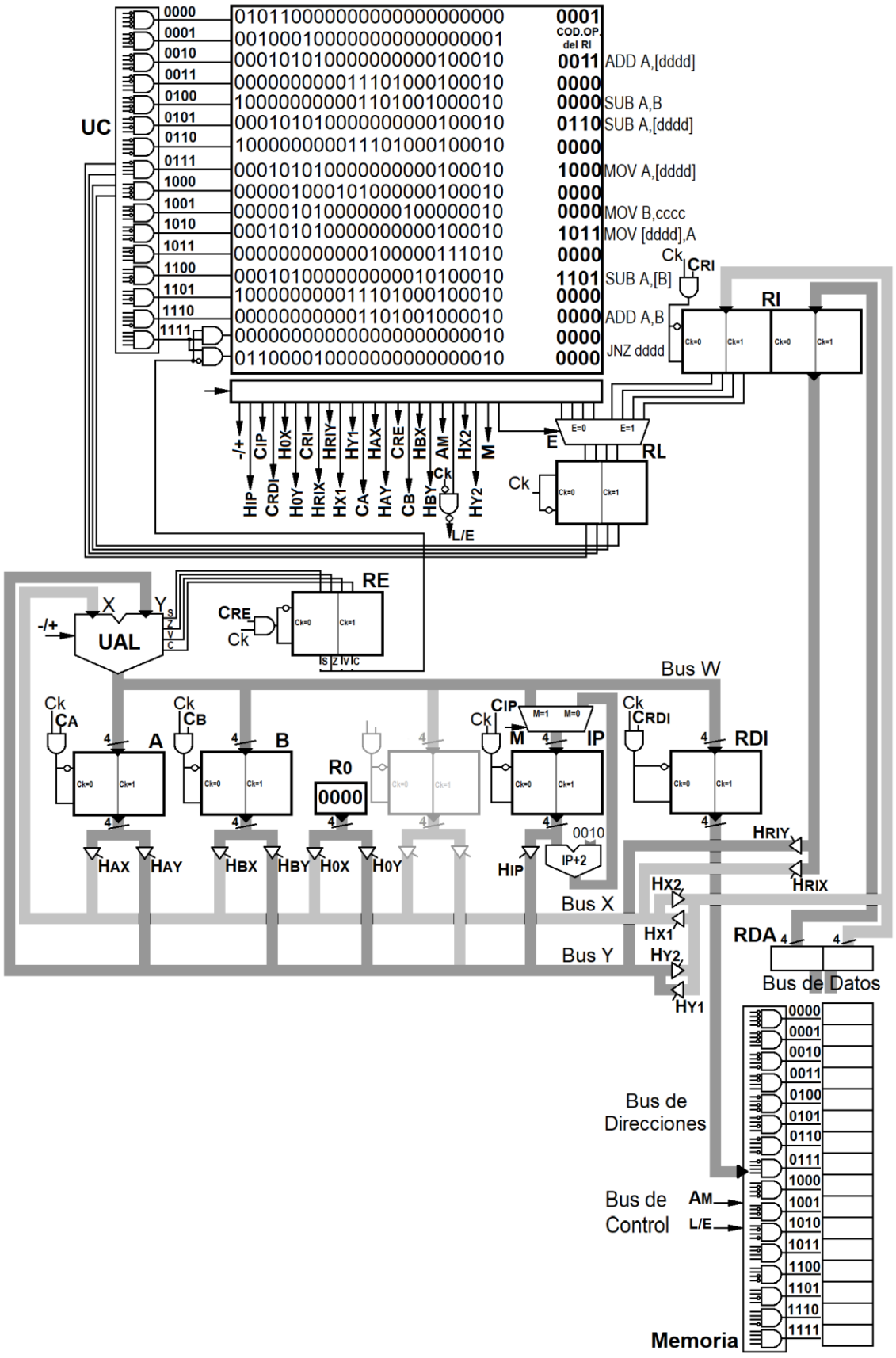
Nombre y Apellido . . . . . DNI . . . . . Curso . . . . . .

TRABAJO PRÁCTICO N° 1
CIRCUITOS LÓGICOS – MODELO DE PROCESADOR

1) Construir la tabla de funcionamiento de un circuito que reconozca sólo la combinación 011 y dibujar el circuito correspondiente.
2) Construir la tabla de funcionamiento de un circuito **de 2 salidas** que reconozca sólo las combinaciones 010 y 011 (una salida para reconocer cada combinación). Construir el circuito correspondiente. Verificar que el circuito cumple con la tabla para 010 y 110.
3) Indicar la tabla de funcionamiento de un circuito que reconozca cada una de las combinaciones posibles de 3 bits. Construir el circuito correspondiente. Verificar que cumple con la tabla para 010 y 110. ¿Cómo se denomina este circuito y cómo se usa en la memoria y en la UCP?
4) Construir la tabla de funcionamiento de un circuito **con una sola salida**, que valga 1 en caso de que en las entradas se activen **cualquiera** de las dos siguientes combinaciones (010 **ó** 011). Construir el circuito correspondiente. Verificar que se cumple con la tabla para las entradas 010 y 110.
5) Dada la tabla de una X-OR construir un circuito con AND y OR que cumpla con dicha tabla.
6) Construir la tabla de funcionamiento de un circuito que en sus 4 entradas reciba dos números cualesquiera de 2 bits, y que la combinación binaria que aparece en sus 4 salidas corresponda al producto de esos dos números presentes en sus entradas. **Construir el circuito** y verificar para 11 x 10
7) Construya una UAL para sumar o restar dos números de 4 bits y que genere los flags luego de cada operación. Luego pruebe el circuito con las siguientes operaciones con números enteros, y determine circuitalmente el valor de los flags, previa realización de las cuentas en papel: (– 7) + (– 3) ; (– 7) – (– 3) .
8) Explique circuitalmente cómo es que el flag C no se invierte en la suma y se invierte en la resta.
9) Explique circuitalmente que si la salida que da el valor del flag V vale 1, se corresponde con el hecho de que se suman dos naturales que representan a enteros de igual signo y el resultado es un natural que representa un entero de signo contrario.
10) Dibujar un flip flop "D latch" con sus tres compuertas. Luego analizar cada una de las siguientes dos secuencias
    I. a) con Ck=0 está reteniendo un cero. b) con Ck=1 se prepara para guardar un uno. c) con Ck=0 terminó guardando un cero.
    II. a) con Ck= 0 está guardando un cero. b) con Ck=1 se prepara para guardar un 1. c) termina guardando un uno. ¿Qué supone que ocurrió en la primera secuencia?
11) Para los instantes del punto 10) dibujar los diagramas temporales de Ck, D y Q.
12) Explicar por qué cuando Ck = 1 el circuito se comporta como un cable entre Q y D y no retiene
13) Dada en el pizarrón la siguiente variación de Ck y D determinar en el tiempo cómo variará Q:



14) Dado un registro con 2 flip flops M-E que cuando Ck = 0 guarda 10 y recibe del exterior 11. Dibujar el registro completo con sus 4 flip flops (2 Maestros – 2 Esclavos), indicando cuáles retienen, cuáles copian y el valor 1/0 de los cables. Idem suponiendo que Ck = 1 y que luego vuelve a ser Ck = 0 recibiendo del exterior 00.
15) Mediante 3 hojas del modelo de procesador en blanco, describir el pedido y la ejecución de una instrucción con código de operación 1111, que se encuentra en la dirección 1010 y ordena saltar a 0111 en el caso de que el flag Z=1. Determinar el valor del dato asociado y en cada hoja indicar los movimientos de información y qué líneas de control deben activarse para permitir dichos movimientos. Indicar valores en buses, UAL y registros, completar en los registros que cambian, los valores de Maestros y Esclavos, tanto para el valor cero del Clock, como para el valor uno.


UC
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
0101100000000000000000000 0001 COD.OP. del RI
0010001000000000000000001
0001010100000000000100010 0011 ADD A,[dddd]
0000000000011101000100010 0000
1000000000011010010000010 0000 SUB A,B
0001010100000000000100010 0110 SUB A,[dddd]
1000000000011101000100010 0000
0001010100000000000100010 1000 MOV A,[dddd]
0000010001010000000100010 0000
0000010100000000100000010 0000 MOV B,cccc
0001010100000000000100010 1011 MOV [dddd],A
0000000000001000001110 0000
0001010000000000010100010 1101 SUB A,[B]
1000000000011101000100010 0000
0000000000011010010000010 0000 ADD A,B
0000000000000000000000010 0000
0110000100000000000000010 0000 JNZ dddd
-/+ CIP H0X CRI HRIX HY1 HAX CRE HBX AM HX2 M
HIP CRDI H0Y HRIX HX1 CA HAY CB HBY HY2
L/E
Ck
CRI
RI
E
RL
RE
UAL
X
Y
S
Z
V
C
CRE
Bus W
CA
A
CB
B
R0
0000
CIP
M
IP
CRDI
RDI
HAX HAY HBX HBY H0X H0Y
HIP
0010
IP+2
HRIY
HRIX
HX2
HX1
HY2
HY1
Bus X
Bus Y
RDA
Bus de Datos
Bus de Direcciones
Bus de Control
AM
L/E
Memoria

UNIVERSIDAD ABIERTA INTERAMERICANA
FACULTAD DE TECNOLOGÍA INFORMÁTICA

Asignatura: Sistemas de Computación II

Nombre y Apellido . . . . . DNI . . . . . Curso . . . . . .

TRABAJO PRÁCTICO N° 2
Práctica de Codificación y Ejecución de programas en Assembler

Este trabajo consiste en codificar y ejecutar un conjunto de programas en Assembler, tomados del libro "Assembler desde Cero".

Para presentar el trabajo, el alumno utilizará las cifras de su DNI, para determinar la dirección de inicio de los programas y la dirección de inicio de las áreas de datos. El alumno adjuntará al trabajo una copia de su DNI.

Utilizando como ejemplo el DNI 12.345.678, el programa comenzará en 5678h y el área de datos en 1234h.

Los programas a codificar son los desarrollados en los siguientes Ejercicios del libro:

Ej. 4 Multiplicación mediante Sumas repetitivas (Integra los ejercicios: 2-3-4 y 5)
Ej. 6 Realización de una Suma en 32 bits
Ej. 7 Movimiento de un Vector
Ej. 8 Suma de todos los valores de un Vector (Integra los ejercicios 8-10 y 11)
Ej. 9 Búsqueda en una lista
Ej. 12 Búsqueda del mayor número de una lista
Ej. 13 Separar una lista en dos listas correspondientes a los negativos y los positivos respectivamente
Ej. 14 Orden alfabético de listas delimitadas por el carácter 0Dh
Ej. 18 Uso de indicadores de pasada
Ej. 22 Verificación de la memoria.

Se codificará de la forma más efectiva para cumplimentar la tarea y luego se ejecutarán los programas paso a paso, analizando la acción de cada una de las instrucciones.
El valor de la variable que en el registro Cx determina las veces que se hará un bucle se elegirá con valores entre 2 y 4 , para evitar tener que ejecutar dicho bucle demasiadas veces.

A cada programa se le agregará una descripción donde se detalla el uso de cada registro y de cada flag.

Se analizará el resultado esperado de cada programa antes de su ejecución, para verificar al concluir la misma que el programa fue correctamente codificado.

Se utilizará DEBUG, ya sea sobre una máquina virtual o sobre DOSBOX. Se utilizará la siguiente secuencia de comandos:

1. Comando **E**, para entrar los datos elegidos (direcciones y valores elegidos del DNI)
2. Comando **A**, **dirección inicial** para editar el programa en Assembler (dirección inicial elegida de Las últimas 4 cifras del DNI), cuya ultima instrucción será INT 20 o INT21 con AH=4C.
3. Comando **U**, **dirección inicial** para corroborar que el programa ha sido bien escrito.
4. Comando **RIP** , para que el IP apunte a la dirección de la primer instrucción.
5. Comando **R**, para verificar que la primera instrucción a ejecutar sea la correcta.
6. Sucesión de comandos **T** hasta que la próxima instrucción a ejecutar sea INT 20, la cual no debe ser ejecutada, antes de que se pierda el programa editado en pantalla.
7. Cada vez que se ejecute un comando T se debe observar que los registros que deben cambiar lo hagan de la forma esperada, y que la próxima instrucción sea la correcta.

Como se observa, la única diferencia importante respecto de lo realizado en Sistemas de Computación I, consiste en que el programa ahora se codifica mediante el comando A, en vez de comando E.

8. Una vez ejecutado el programa. mediante el comando **E** se debe examinar los resultados obtenidos en memoria en las direcciones que corresponda los cuales **deben coincidir con los resultados esperados**, estimados en el paso resaltado previamente. Si esta verificación no se da, se debe encontrar la fuente de error.

9. En la presentación del TP, el alumno, insertará las imágenes de la ejecución paso a paso. Indicará los registros y Flags que cambian, y el por qué se salta o no en las instrucciones de salto condicional.

UNIVERSIDAD ABIERTA INTERAMERICANA
FACULTAD DE TECNOLOGÍA INFORMÁTICA

Asignatura: Sistemas de Computación II

Nombre y Apellido . . . . DNI . . . . . Curso . . . . . .

TRABAJO PRÁCTICO N° 3
Llamado a Subrutinas e Interrupciones por Software.

PARTE I:
Se codificará y ejecutará un programa. Luego se trasladarán instrucciones a una subrutina y se volverá a codificar y ejecutar el nuevo programa que incluirá un llamado a la subrutina. Luego, se convertirá esa subrutina en una rutina a la que se accederá mediante una interrupción por Software. Se codificará y ejecutará un nuevo programa, que mediante una interrupción por software, ejecutará la misma tarea.

0) Para que todos los trabajos sean distintos, el alumno determinará las direcciones de memoria a utilizar mediante su DNI, que deberá fotocopiarse al comienzo del TP.
Tomando como ejemplo el DNI 12.345.678, la dirección de la zona de variables será 1234h, la dirección de la primera instrucción del Programa Principal 5678h, la primera instrucción de la subrutina será 3456h y la dirección de la rutina de la interrupción será 8765:4321

1) Teniendo en cuenta las direcciones anteriores, se realizará el Ej 31 (Convertir una lista de números en ASCII a binario) del libro Asembler desde Cero. Se verificará su funcionamiento y se imprimirá su ejecución para una secuencia de tres caracteres.
2) Ahora se repetirá el programa pero convirtiendo la conversión de ASCII a binario en una subrutina que se ubicará a partir de la dirección calculada en el punto 0.
4) Se estimarán previamente en forma teórica los valores de la pila antes y después de ejecutarse el CALL. Demostrando que durante le ejecución los mismos se cumplen tal cual lo estimado.
5) Se debe imprimir la ejecución, demostrando que se cumple con lo estimado.
6) Una vez concludos los puntos anteriores, se repetirá la ejecución pero convirtiendo la conversión de ASCII a binario, mediante una interrupción por software. INT61
7) Como en todos los casos anteriores, se estimará en forma teórica los cambios en los registros IP, SP, y el contenido del STACK.
8) Para ello usando el Debug se deberá configurar el Vector correspondiente a INT 61 en la Tabla de Vectores de interrupción de la siguiente manera:
a) Previamente con el comando E el alumno escribirá en memoria los valores de los datos y con A el programa y la rutina
b) Luego mediante el comando **E 0000:0184 escribirá en la zona de vectores la dirección de la subrutina del sistema operativo** correspondiente a INT 61
c) Luego se ejecutará el programa comprobando y comprobarán todos los cambios que ocurran en la Pila y en el IP y en el SP, dejando constancia de los mismos en el informe presentado.

PARTE 2:
Ejecución de programas construidos con interrupciones por software que llaman a subrutinas del BIOS.
Tomando como referencia los ejercicios de las pags 77 a 79 del texto “Assembler desde Cero”
Se ejecutarán, usando el comando Gxxxx en vez del comando T:
1) El programa que permite visualizar en pantalla todos los caracteres, teclas y símbolos incluidos en el código ASCII.
2) El programa que se queda esperando que se pulsen teclas del teclado para entrar caracteres.
3) El programa para ingresar el DNI y mostrarlo en pantalla

Ubicar en los tres casos el programa y sus datos de acuerdo al DNI del alumno, y presentar la ejecución de cada programa con una descripción de lo que se realiza.

UNIVERSIDAD ABIERTA INTERAMERICANA
FACULTAD DE TECNOLOGÍA INFORMÁTICA
Sistemas de Computación II

GUÍA DE PREGUNTAS PARA ABORDAR EL APRENDIZAJE

Unidad 1

Hacer el circuito y explicar cómo funciona un decodificador de 3 entradas

Armar un sumador de dos números de 4 bits y mediante tablas de funcionamiento hallar el circuito de cada circuito que suma los 3 bits de cada columna o posición.

Convertir un sumador en un restador y luego armar una UAL con los flags que genera

Establecer la correspondencia entre una suma o resta realizada manualmente y la que realiza el sumador/restador de la UAL

Hacer el circuito y explicar cómo funciona un selector (multiplexor) de 2 entradas

A partir de un selector de 2 entradas construir un flip flop D sincrónico. Suponiendo que dicho flip flop guarda un cero, indicar en el circuito los pasos a seguir para que guarde un uno. Dados los pulsos Ck y la variación en el tiempo de su entrada D determinar la variación de la salida Q

Cómo está constituido un registro

Explicar con un ejemplo, por ej: un sumador asociado al registro IP; por qué los registros de la UCP deben estar constituidos por flip flops dobles "maestro-esclavo"

Dibujar el esquema del registro **RL** de la UCP, y comparar su funcionamiento con el **RDI**.

A partir de los pulsos de clock y de las fotocopias de los circuitos del modelo, describir cómo evolucionan los M y E de los registros en el pedido y ejecución de una instrucción que ordena restar al registro A un dato que está en memoria principal, o una constante, o una instrucción de salto condicional. Suponer valores de direcciones y datos.

Unidad 2

Indicar cuáles son los modos de direccionamiento usados en los TP1 y TP2 de Assembler

Qué se guarda en la pila durante la ejecución de CALL y en qué pasos puede dividirse. Cómo queda la pila. Ejemplificar, indicando también el SP

A qué instrucciones equivale CALL

A qué instrucciones equivale RET

Qué hace una instrucción como PUSH DX, y en qué pasos puede dividirse. Cómo queda la pila. Ejemplificar

Qué hace una instrucción como POP CX y en qué pasos puede dividirse. Cómo queda la pila. Ejemplificar

Unidad 3

Qué es el registro CS, y como con el IP forma la dirección efectiva en modo real.

Qué es una interrupción

Cómo se clasifican las interrupciones por hardware

Dónde se originan las IRQ y por qué causas

Qué pasa con las IRQ que "pierden" si hay varias simultáneamente activas

Qué bloques comunica y qué función cumpla la línea INTR

Cómo se entera el árbitro que debe enviar la dirección del vector

Qué es la zona de vectores interrupción, qué contiene cada uno, y cuando se escribe

Cuándo la UC sensa la línea INTR

Qué es el flag I y cómo se modifica luego de una instrucción INTxx y otra IRET, y qué relación tiene esto con los modos Kernell y User. Cómo aparece esto en el Debug ?

Qué significa enmascarar una interrupción

A partir de qué momento una interrupción por hardware sigue el mismo proceso que una interrupción por software. Indicar este proceso.. Cómo queda la pila. Ejemplificar

Qué hace la instrucción IRET

Para qué sirven las interrupciones tipo IRQ

Para qué sirven las interrupciones tipo software

Describir paso a paso el proceso de una interrupción tipo IRQ

Por qué se dice que una interrupción por software es una auto interrupción

Cómo afectan las interrupciones al programa interrumpido

Qué tienen en común y diferente una INT y un CALL

Unidad 4

Cómo se da la proximidad espacial y temporal en el caché. Ejemplificar con a) instrucciones b) datos

Desarrollar un ejemplo de un caché de 2 vías de 8 bytes con 2 líneas y 8 celdas, suponiendo una situación inicial en correspondencia con dichos valores localizados en memoria principal, incluidos suponer valores de LRU y tags. Luego suponer que direcciones de la UCP producen un hit que cambia dos LRU y después un MISS

Por qué se necesita una jerarquía de memorias

Qué lugar ocupa el caché en la jerarquía de memorias el caché.

Qué implica la denominación caché

Qué es un “wait state” en la UCP

Explicar a partir del 80286 cómo surgieron los dos niveles de caché

Qué porcentaje de aciertos (“hits”) puede estimarse en un caché

Cuáles son las funciones del controlador de caché

Cómo determina el controlador de caché con la tag memory si hay un hit o un miss y qué pasa en cada uno de ellos

En qué caso pasa información de la memoria al caché

Qué es una línea de un caché y a que proximidad se “apuesta” cuando se pide una línea

Explicar el uso de los bits LRU en un caché de 2 vías.

Unidad 5

Explicar qué es el pipeline y qué beneficios reporta para la UCP.

Qué significa que el Pentium sea un procesador super escalar, qué debe tener en su interior para ello, y en qué caso funciona de esa manera

Qué tipos de instrucciones similares y diferentes tienen un CISC y un RISC

Por qué un RISC no tiene ROM de Control, y qué ventajas ello implica

Cuáles son las características generales de un procesador RISC y que ventajas le confieren frente a un RISC

Cuáles son las características generales de un procesador hyper threading o dual core actual

Qué diferencia existe entre multiprocesamiento y multiprogramación

Unidad 6

Indicar el papel de los ports en la activación de señales, y qué loop de 3 instrucciones se queda efectuando la UCP

Cuáles son en detalle las 5 fases de una operación de escritura de un rígido por ADM

Describir **detalladamente** qué pasa en la fase de preparación (integrando Directorio Raiz, FAT, clusters, CHS, etc.)

Cuándo ocurre la sincronización, y qué señal la indica

Mediante qué instrucciones se escribe un port

Mediante qué instrucciones se lee un port

En qué fases interviene la UCP en una lectura o escritura por ADM

Qué ventajas tiene trabajar por ADM

Cuántos ADM y escrituras deben hacerse para pasar de memoria a memoria externa, y de qué depende

Cuándo comienza la fase de verificación, y qué ocurre durante la misma

Comparar E/S por ADM con E/S con PIO

Indicar cómo se hace la sincronización y la transferencia en PIO

Indicar cómo sería la lectura de un sector usando PIO

Cuántos sectores se leen o escriben en cada E/S

Describir la secuencia de pasos que suceden desde que se pulsa una tecla hasta que la misma se visualiza en pantalla

UNIVERSIDAD ABIERTA INTERAMERICANA
FACULTAD DE TECNOLOGÍA INFORMÁTICA
Sistemas de Computación II

GUIA DE ABORDAJE BIBLIOGRÁFICO

Se propone la siguiente bibliografía, las tres primeras obras de forma obligatoria, las tres subsiguientes optativas.

Se indican en la tabla los capítulos o títulos de las obras que tratan los distintos puntos del programa.

En el caso de indicarse una "p" al lado del número, es porque el tratamiento del punto del programa en esa referencia es de tipo "parcial".

La bibliografía es:

I La Pc por Dentro (6ta.edición) Mario C. GINZBURG

II Periféricos (4ta. edición) Mario C. GINZGURG

III Assembler desde cero (3a. edición) Mario C. GINZBURG

IV De la compuerta al computador (1a. edición) Mario C. GINZBURG

V La Pc por Dentro (4ta.edición) Mario C. GINZBURG

VI Principios de arquitectura de computadoras (1a. edición) Vincent P. HEURING – Miles MURDOCCA

VII Organización y arquitectura de computadoras (7a. edición) William STALLINGS

VII Organización de computadoras – Un enfoque estructurado (4ta. edición) Andrew S. TANEMBAUN

		I	II	III	IV	V	VI	VII	VIII
1	Señales digitales.				1		Ap		
2	Compuertas.				1		A4	B2	3.1.1
3	Decodificadores.				1		7.8.1 A10	B3	3.2.2
4	Multiplexores.				1		A10	B3	3.2.2
5	Sumadores.				1		Ap	B3p	3.2.3
6	UAL.				1			B3p	3.2.3p
7	Flip-flop R-S, Flip-flop "D" latch y Flip-flop maestro-esclavo.				1		A11	B4p	3.3.2p
8	Registros.				1		A14P	B4p	3.3.3p
9	Memoria.				A		7p	Bp	3.3.4
10	Modelo didáctico de computador basado en el Modelo desarrollado en Sistemas de Computación I, integrando los circuitos antes estudiados. Pedido y ejecución de instrucciones en diversos modos de direccionamiento en el Modelo.						6p A12p	16p 17p	
11	Aplicación a las instrucciones de salto condicionado.	1.10							

12	Programación en Assembler usando el Debug.			1			4p	15p	
13	Modos básicos de direccionamiento.			1			4.5p	11.1	5p
14	Manejo de Vectores.			1					
15	Llamados a subrutinas.			1				10.5p	
16	Uso de la pila y del stack pointer.			1			4.6	10.5p	
17	Interrupciones por software y por hardware y su relación con el sistema operativo.			1				7.4p	
18	Tabla de Vectores de Interrupción.			1					
19	Tipo de interrupciones por hardware.			1					
20	Enmascaramiento de interrupciones originadas en líneas tipo IRQ			1					
21	Memoria caché.					1.12	7.6	4	2.2.5p 4p
22	Proximidad temporal y espacial en el caché Niveles.					1.12	7.6		4.5.1p
23	Caché de correspondencia directa y de 2 vias.					1.12	7.6	4	4.5.1p
24	Memorias asociativas.					1.12	7.6	4	4.5.1p
25	Funciones del controlador de caché.					1.12	7.6	4	
26	Jerarquía de memorias.					1.12	7.1	4.1	2.3.1
27	Pipeline en el 486, Pentium y procesadores actuales.					1.14	10p	14	3.5p
28	Procesamiento interior.					1.14		14	4p
29	Ejecución fuera de orden.					1.14		14	4p
30	Procesadores CISC y RISC.					1.14	10.2p	13	2p
31	Hyper Threading y multicore.					1.14	10p	18.4p	4p
32	Características de los procesadores RISC.					1.14	10.3p	13	4p
33	Detalle de interconexiones entre memoria y periféricos a través del Northbridge y Southbridge.	1.7 1.8					8.2 p		
34	Interconexionado entre los buses PCI, SCSI y USB y PCI.	1.7 1.8				1.13	8.2p		2.3.5p 3.6p

35	Ports.	1.7							
36	Fases en las operaciones de E/S.		2.1					7.3p	
37	Instrucciones IN/OUT		2.1	1					
38	Interfaz port paralelo.			1					
39	Casos del disco (ADM y PIO), disquete (detalles de las 5 fases en ADM), impresora (con port paralelo) y teclado –		2.1				8.3p	7.5p	
40	Transferencias por ADM y AIM (PIO).		2.1				8.3p		
41	Canal controlador de ADM y sus registros.		2.1						5.5.7p
42	Ventajas del ADM sobre el PIO.		2.1						

UNIVERSIDAD ABIERTA INTERAMERICANA
FACULTAD DE TECNOLOGÍA INFORMÁTICA
Sistemas de Computación II

MAPA CONCEPTUAL DE LA MATERIA