

# ESTRUCTURA DE COMPUTADORES

GRADO EN INGENIERÍA INFORMÁTICA

DOBLE GRADO EN INGENIERÍA INFORMÁTICA Y  
ADMINISTRACIÓN DE EMPRESAS

## **Práctica 2**

## **Introducción a la microprogramación**



UNIVERSIDAD CARLOS III DE MADRID

Grupo de Arquitectura de Computadores, Comunicaciones y Sistemas

Curso 2019/2020

Versión 2.5

# Contenido

Objetivos de la práctica .....	3
Ejercicio 1 .....	4
Ejercicio 2 .....	7
Procedimiento de entrega de la práctica.....	9
Aspectos importantes a tener en cuenta.....	10
Normas generales .....	10
Códigos de la práctica.....	10
Memoria de la práctica .....	11
Puntuación en la práctica .....	11
Apéndice 1: Grabación de prueba.....	13
Apéndice 2: Guardar un <i>checkpoint</i> en WepSIM .....	15

## Objetivos de la práctica

El principal objetivo de esta práctica es aprender a diseñar un juego de instrucciones para un computador.

Los principales conocimientos que se practicarán son los de microprogramación, programación en ensamblador y representación de la información. También se aprenderá a evaluar distintas alternativas de diseño y a trabajar en grupo.

Para el desarrollo de la práctica, es necesario repasar los siguientes conceptos:

- La representación de números enteros, cadenas de caracteres, etc.
- Los principales aspectos del lenguaje ensamblador.
- El formato de instrucciones y los tipos de direccionamiento.
- El funcionamiento de un procesador, incluyéndose las etapas de ejecución, microprogramación, etc.

El/La estudiante **utilizará** el simulador WepSIM para poder ejercitar de una forma interactiva los conceptos y conocimientos anteriormente indicados, completando de esta forma los ejercicios de la asignatura y mejorando asimismo el conocimiento del funcionamiento global de un computador.

El simulador se encuentra disponible en <https://wepsim.github.io/wepsim> y la documentación inicial es accesible desde <https://wepsim.github.io>.

## Ejercicio 1

La empresa para la que trabajamos nos pide **diseñar, implementar y probar** el conjunto de instrucciones adaptadas del Z80 para el simulador WepSIM indicadas en la tabla 1. Las instrucciones se codificarán en 32 bits.

Instrucción	Formato	Funcionalidad asociada	Registro de estado
ld R <sub>DEST</sub> , R <sub>ORIG</sub>	CO (31-26): 010000 R <sub>DEST</sub> (25-21) R <sub>ORIG</sub> (20-16)	BR[R <sub>DEST</sub> ] ← BR[R <sub>ORIG</sub> ]	<u>No</u> se actualiza
ldi R <sub>DEST</sub> , U16	CO (31-26): 010010 R <sub>DEST</sub> (25-21) U16 (15-0)	BR[R <sub>DEST</sub> ] ← U16	<u>No</u> se actualiza
ld R <sub>DEST</sub> , (R <sub>ORIG</sub> )	CO (31-26): 010011 R <sub>DEST</sub> (25-21) R <sub>ORIG</sub> (20-16)	BR[R <sub>DEST</sub> ] ← Memoria[R <sub>ORIG</sub> ]	<u>No</u> se actualiza
add_a R <sub>ORIG</sub>	CO (31-26): 011000 R <sub>ORIG</sub> (25-21)	A ← A + BR[R <sub>ORIG</sub> ]	Se actualiza
addi_a S16	CO (31-26): 011010 S16 (15-0)	A ← A + S16	Se actualiza
inc R <sub>DEST</sub>	CO (31-26): 011100 R <sub>DEST</sub> (25-21)	BR[R <sub>DEST</sub> ] ← BR[R <sub>DEST</sub> ] + 1	Se actualiza
dec R <sub>DEST</sub>	CO (31-26): 011101 R <sub>DEST</sub> (25-21)	BR[R <sub>DEST</sub> ] ← BR[R <sub>DEST</sub> ] - 1	Se actualiza
jp S16	CO (31-26): 110000 S16 (15-0)	PC ← PC + S16	<u>No</u> se actualiza
jpz S16	CO (31-26): 110011 S16 (15-0)	Si (Flag.Zero == 1) PC ← PC + S16	<u>No</u> se actualiza
call U16	CO (31-26): 100001 U16 (15-0)	SP ← SP - 4 Memoria[SP] ← PC PC ← U16	<u>No</u> se actualiza
ret	CO (31-26): 100010	PC ← Memoria[SP] SP ← SP + 4	<u>No</u> se actualiza
halt	CO (31-26): 100011	PC ← 0x00	<u>No</u> se actualiza
push R <sub>ORIG</sub>	CO (31-26): 100100 R <sub>ORIG</sub> (25-21)	SP ← SP - 4 Memoria[SP] ← R <sub>ORIG</sub>	<u>No</u> se actualiza
pop R <sub>DEST</sub>	CO (31-26): 100101 R <sub>DEST</sub> (25-21)	R <sub>DEST</sub> ← Memoria[SP] SP ← SP + 4	<u>No</u> se actualiza

Tabla 1.- Conjunto de instrucciones de Z80

Se usará R<sub>DEST</sub> y R<sub>ORIG</sub> para denotar los registros de propósito general del Z80, que en esta versión de 32 bits son: A, BC, DE, HL, IX y IY. Se usará BR para referenciar al Banco de Registros, y BR[R<sub>DEST</sub>] para indicar el contenido del registro R<sub>DEST</sub>. Los números enteros almacenados en los registros son de 32 bits usando complemento a dos.

Los valores “S16” indican que hay que hacer extensión de signo mientras que en los “U16” no se hace extensión de signo (se rellena con ceros a la izquierda).

A continuación se indica la asociación entre los registros del Z80 y los registros de WepSIM. Esta asociación tiene que indicarse en la sección de registros del microcódigo de las instrucciones pedidas.

Registro Z80	Registro WepSIM	Significado
A	R4	Acumulador
BC	R5	Registro compuesto BC
DE	R6	Registro compuesto DE
HL	R7	Registro compuesto HL
IX	R8	Registro compuesto IX
IY	R9	Registro compuesto IY

El registro puntero de pila (SP) es el R29 del procesador elemental de WepSIM, y el registro de *Flags* es el registro SR en el procesador de WepSIM. Flag.Zero hace referencia al bit Z (cero) del registro de estado del procesador WepSIM.

Para pasar parámetros a una subrutina en el ensamblador del Z80 se usan los registros IX e IY (tienen la misma funcionalidad que los registros \$a0 y \$a1 en MIPS) y para devolver un resultado se usa el registro HL (equivalente al \$v0 en MIPS). En caso de pasar más de dos argumentos a una función, el tercero y el resto de los argumentos se pasarían en la pila.

Se valorará que se haya realizado una implementación válida que minimice el número de ciclos de reloj, justificando brevemente en la memoria las decisiones de diseño que se han tomado para lograrlo.

Los resultados de este ejercicio son relativos al diseño y microcódigo, y se indicarán tanto en la parte de la memoria correspondiente como en el fichero asociado con la funcionalidad pedida.

La sección en la memoria para el ejercicio 1 ha de contener:

- Una tabla con cuatro columnas. La primera columna incluye el nombre de la instrucción, la segunda el diseño de las instrucciones pedidas en lenguaje RT. En esta segunda columna hay que indicar, por cada ciclo, las operaciones elementales de transferencia entre registros, necesarias para la ejecución de la instrucción. La tercera incluye las señales de control que se han de activar en cada ciclo de cada instrucción. La cuarta incluye las decisiones de diseño que haya tomado para esta instrucción. Esta tabla tendrá tantas filas como instrucciones se han pedido en el enunciado.

El fichero con la funcionalidad pedida es:

- **e1\_checkpoint.txt:**

Se ha de guardar un *checkpoint* que contenga el microcódigo de las instrucciones pedidas (vea apéndice 2). **Solo ha de incluir el microcódigo correcto para las instrucciones pedidas (sin fetch)** según los requisitos dados y estando en el formato correcto para el simulador WepSIM.

## Ejercicio 2

Para probar las instrucciones puede codificar los programas que considere apropiados. No obstante, la empresa nos pide, para poder hacer una demostración, que realicemos un programa en ensamblador, que use el conjunto de instrucciones del Z80 pedido en el ejercicio 1, de forma que dicho programa permita también asegurar el cumplimiento de los requisitos pedidos (funcionalidad descrita en el ejercicio 1).

El programa a codificar, utilizando las instrucciones del Z80 diseñadas en el apartado anterior, tendrá la misma funcionalidad que la del siguiente programa escrito en el ensamblador MIPS32:

```
.data
    vector: .word 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

.text
sumav: # apilar $a0 y $a1
        addi $sp $sp -8
        sw $a0 4($sp)
        sw $a1 0($sp)
        # $v0 = suma de los elementos de vector
        li $v0 0
b1:     beq $a0 $0 f1
        lw $t0 ($a1)
        add $v0 $v0 $t0
        addi $a1 $a1 4
        addi $a0 $a0 -1
        b b1
        # desapilar $a1 y $a0
f1:     lw $a1 0($sp)
        lw $a0 4($sp)
        addi $sp $sp 8
        # return
        jr $ra

main:   # llamar a la subrutina contar
        li $a0 10
        la $a1 vector
        jal sumav
        # terminar la ejecución
        li $v0 10
        syscall
```

Este programa tiene una rutina **sumav** que recibe dos parámetros: el número de elementos de un vector de enteros y la dirección de comienzo de dicho vector de enteros. La rutina devuelve en \$v0 la suma de los números contenidos en el vector pasado por parámetro. También hay una rutina **main** que se encarga de llamar a la

rutina **sumav** y de terminar el programa. Tenga en cuenta los registros usados en el convenio de paso de parámetros para el Z80 y que está descrito en la página 5 de este enunciado.

Los resultados de este ejercicio han de indicarse tanto en la parte de la memoria correspondiente a este ejercicio como en el fichero asociado con la funcionalidad pedida.

La sección en la memoria para el ejercicio 2 ha de contener:

- Tras codificar el programa pedido en las instrucciones de Z80 del ejercicio 1 con los mismos nombres para las rutinas y vector que el programa MIPS32, ha de comparar ambos juegos de instrucciones indicando: diferencias de los tipos de instrucciones, ventajas e inconvenientes que presentan cada uno de dichos juegos de instrucciones.

El fichero con la funcionalidad pedida es:

- **e2\_checkpoint.txt:**

Se ha de guardar un *checkpoint* que contenga el **programa en ensamblador** de la prueba pedida en el ejercicio 2, según los requisitos dados y estando en el formato correcto para el simulador WepSIM.

El *checkpoint* ha de contener, junto con el programa en ensamblador, **una grabación** que ayude a probar que el microcódigo del ejercicio 1 es correcto y que se ha probado. Al menos antes de la llamada **sumav** se ha de indicar los resultados esperados y tras la llamada se ha de explicar los resultados obtenidos.

El apéndice 1 resume los pasos para realizar una grabación. El apéndice 2 resume los pasos para guardar un *checkpoint*.



## Procedimiento de entrega de la práctica

La entrega de la práctica 2 se realizará de forma electrónica a través de Aula Global, para lo que se habilitará dos entregadores asociados a dicha práctica.

La fecha límite de entrega es el día **2 de diciembre de 2019 a las 23:50 horas**.

Es posible entregar tantas veces como quiera dentro del plazo dado, la única versión registrada de su práctica es la última entregada. La valoración de la práctica es la valoración del contenido de esta última entrega. Por favor revise (usted y resto del grupo) siempre lo que entregue.

### A entregar:

Un entregador se habilita para entregar la memoria a través de turnitin. Se ha de entregar un fichero en formato PDF con el nombre AAAAAAAAAA\_BBBBBBBBBB.pdf donde A...A y B...B son los NIA de los integrantes del grupo.

El otro entregador es para entregar todo en un único archivo comprimido en formato **zip** con el nombre AAAAAAAAAA\_BBBBBBBBBB.zip donde A...A y B...B son los NIA de los integrantes del grupo.

El archivo **zip** debe contener solo los siguientes archivos (sin subdirectorios):

- **e1\_checkpoint.txt**
- **e2\_checkpoint.txt**
- **memoria.pdf**
- **autores.txt**

Donde el fichero “autores.txt” contendrá una línea por autor con solo su NIA correspondiente.

**Todos los archivos serán de texto tipo ASCII salvo la memoria que será en formato PDF.** La memoria es la misma que la entregada a través del entregador turnitin habilitado para la entrega de la memoria únicamente. Compruebe que ha usado cada entregador correctamente para evitar que la entrega sea nula (y por tanto, de igual valor que no haberla entregado).

## Aspectos importantes a tener en cuenta

**Ha de revisar cuidadosamente que se cumple todos los requisitos indicados en el enunciado.** Se recomienda a partir del enunciado generar una lista de comprobación (*checklist*) que ayude al grupo de práctica a revisar todo.

**Es importante recordar que el nombre de las subrutinas, ficheros, etc. han de ser los indicados en este enunciado.** No respetar dichos nombres supondrá un cero de nota en el apartado correspondiente por lo que se recomienda añadir a la lista de comprobaciones las comprobaciones de los nombres.

### ***Normas generales***

- 1) La entrega de la práctica se realizará a través de los entregadores habilitados. No se permite la entrega a través de correo electrónico.
- 2) La entrega se realizará en el plazo dado por los entregadores. Es posible que para un entregador de Aula Global el fin del plazo para una entrega a las 24:00 termine 10 minutos antes. Revise el soporte de Aula Global para confirmar el plazo.
- 3) Se prestará especial atención a detectar funcionalidades copiadas entre prácticas. En caso de detectar copia en dos prácticas (o en la memoria), todos los grupos implicados (copiados y copiadore) obtendrán una calificación de 0 (cero), pudiendo abrirse un expediente dependiendo de la gravedad. La copia de porciones de Internet o de prácticas de otros cursos es también considerado copia, y todos los grupos implicados podrán ser expedientados.

### ***Códigos de la práctica***

- 1) Se valorará que se haya realizado una implementación válida que minimice el número de ciclos de reloj.
- 2) Un programa no comentado adecuadamente obtendrá una calificación de 0. Los comentarios han de buscar describir cada paso que se pretenden implementar antes del bloque de código que lo implementa.
- 3) Hay que tener en cuenta que un programa que compile correctamente no es garantía de que funcione correctamente. Por ello tendrá que realizar aquellas pruebas que garanticen el correcto funcionamiento de la práctica.
- 4) En el código pedido para entregar **no** han de **imprimirse mensajes** por pantalla ni contener ningún otro código adicional usado para diagnóstico **que no sea el descrito en el enunciado**.
- 5) Todos los ejercicios han de funcionar en todo momento con la versión del simulador WepSIM dada en la URL: **<https://wepsim.github.io/wepsim/>**

## ***Memoria de la práctica***

- 1) **La longitud de la memoria no deberá superar las 12 páginas** (portada e índice incluidos).
- 2) **Se entregará en formato PDF con texto seleccionable** (ha de permitir que turnitin pueda realizar el control de copia en la memoria).
- 3) La memoria (un único documento) tendrá que contener al menos los siguientes apartados:
  - Portada donde figuren los autores (incluyendo nombre completo, NIA y grupo de cada autor), titulación, asignatura y práctica.
  - Índice de contenidos.
  - Sección para el ejercicio 1 donde se incluya lo pedido para dicho ejercicio.
  - Sección para el ejercicio 2 donde se incluya lo pedido para dicho ejercicio.
  - Conclusiones y problemas encontrados. Incluya un resumen del número de horas destinada a la práctica.

### **NOTA: NO DESCUIDE LA CALIDAD DE LA MEMORIA DE SU PRÁCTICA.**

Aprobar la memoria es tan imprescindible para aprobar la práctica, como el correcto funcionamiento de la práctica. Si al evaluarse la memoria de su práctica se considera que no alcanza el mínimo admisible, su práctica estará suspensa.

## ***Puntuación en la práctica***

La puntuación de la práctica se repartirá entre el código y memoria entregada de la siguiente forma:

- **Microcódigo y código (7 puntos)**
- **Memoria (3 puntos)**

### **MUY IMPORTANTE:**

- 1. Aunque la puntuación se pueda repartir en apartados, la corrección de la práctica se realizará a nivel global, es decir, esto incluye:**
  - a. Si un ejercicio no se entrega la calificación de toda la práctica será de cero.**
  - b. Si se detecta un error de concepto grave en la práctica (en cualquier apartado de cualquier ejercicio), la valoración global de toda la práctica será de cero puntos (0 puntos).**
- 2. Se realizarán comprobaciones para evitar copias totales o parciales en trabajo entregado, es decir, esto incluye:**
  - a. En caso de encontrar implementaciones comunes en dos prácticas (o contenidos similares en la memoria), ambas obtendrán una calificación de 0.**
  - b. En caso de encontrarse fragmentos de código obtenidos directamente de Internet no referenciados, la práctica tendrá una calificación de cero.**
- 3. Ha de respetar el formato y nombres pedidos, esto incluye:**
  - a. Es fundamental respetar el nombre y formato de los ficheros puesto que en caso contrario supondrá una nota de 0 (cero). Ello incluye no respetar minúsculas, entregar un .rar en lugar de .zip, un .docx en lugar de pdf (no vale renombrar), etc.**
  - b. El texto del archivo con la memoria (memoria.pdf) ha de ser seleccionable y copiable. Es decir, si se genera un archivo PDF basado en una imagen por página (para evitar su tratamiento por las herramientas de control de copia) entonces la calificación será de un cero para toda la práctica.**




## Apéndice 1: Grabación de prueba

Las pruebas son clave para demostrar que el microcódigo funciona correctamente, de acuerdo con las especificaciones dadas. El coste de tener que reprogramar millones de procesadores es muy alto y por ello el microcódigo ha de ser revisado meticulosamente.


Realizar las primeras pruebas mediante la ejecución paso a paso ayuda a entender el efecto en la circuitería y será de gran ayuda para aprender. Una prueba consiste en una ejecución de un código de prueba y la comprobación de que en distintos puntos de la ejecución es el estado esperado en cada punto es el estado obtenido.

Una vez se comprenda el funcionamiento de los distintos elementos en un microprograma, es posible seguir realizando las pruebas de una forma más automatizada. Para ello WepSIM permite grabar la mayor parte de las interacciones con la interfaz, así como añadir en distintos puntos de la ejecución comentarios. Se aconseja antes de hacer una grabación ensayar previamente los pasos que se quieren grabar.



### Grabación de sesión de trabajo

Para trabajar con grabaciones ha de pulsar el botón 'RecordBar' arriba en la pantalla.	
Debajo de la pantalla aparecerá la barra de herramientas para las grabaciones.	
Para empezar la grabación de una sesión ha de pulsarse el botón 'Record'. Los principales pasos siguientes con WepSIM serán grabados.	
Cuando se termine la sesión de trabajo a grabar ha de pulsar el botón de 'Stop'.	


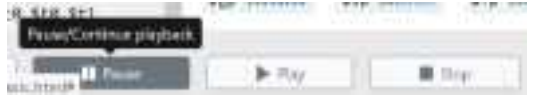

### Borrado de sesión de trabajo

Si desea borrar la grabación (para realizar otra) ha de pulsar el botón 'Reset'. Aparecerá un cuadro de diálogo que le pedirá confirmar que quiere borrar (botón 'Reset' de nuevo).	
---	--

## Añadir un comentario a una grabación

<p>Mientras se está grabando, es posible añadir un comentario a la grabación. Para ello pulse el botón 'Comment'.</p>	
<p>Aparece un cuadro de diálogo para rellenar los datos asociados al mensaje: título, contenido del mensaje y duración (en segundos) del mensaje.</p>	

## Reproducción de sesión de trabajo




<p>Para reproducir una grabación ha de pulsar el botón 'Play'.</p>	
<p>Puede pausar la reproducción de la sesión con el botón 'Pause'. Al volver a dar al botón de 'Pause' continua la ejecución.</p>	
<p>Para detener y volver al paso inicial ha de pulsar el botón 'Stop'.</p>	

## Apéndice 2: Guardar un *checkpoint* en WepSIM

WepSIM permite guardar en un solo fichero toda la sesión de trabajo. Dicha sesión puede incluir el microcódigo pedido, el código en ensamblador, los estados en distintos puntos de ejecución y una grabación de la sesión de trabajo. De esta forma es más ágil continuar el trabajo o compartir dicho trabajo entre integrantes del grupo de práctica.

Para ello se ha de usar lo que en el simulador WepSIM se llama *checkpoint*. A continuación se muestra los pasos para guardar un checkpoint.

### Guardar un *checkpoint*

<p>En el menú de modo de ejecución seleccione la opción “<i>Checkpoint</i>”.</p>	 A screenshot of the WepSIM web application. The top navigation bar has 'Examples', 'Help', and 'EP'. The 'Execution' menu is open, showing options: 'WepSIM hardware', 'EP', 'Pick firm/soft from:', 'Examples', 'Checkpoint' (highlighted with a red arrow), 'Information from:', 'Help', 'Notifications', and 'Welcome tutorial'. The background shows a circuit diagram with components like 'MUX A', 'MUX B', 'ROM A', and 'ROM B'.
<p>Indique el nombre del fichero en el campo “File name:” y a continuación pulse el botón “Save” para guardar el fichero.</p>	 A screenshot of the 'Checkpoint' dialog box. It has two tabs: 'Output' and 'Input'. The 'Output' tab is active, showing a 'File name' input field with 'ad_checkpoint' entered, and a 'Save' button. The 'Input' tab is empty. A red arrow points to the 'Save' button.
<p>Compruebe que el fichero se ha guardado correctamente y contiene todo lo pedido en el enunciado.</p>	 A screenshot of a file explorer window showing the saved checkpoint file. The file is named 'ad_checkpoint' and is located in the 'Downloads' folder. The file size is 100 KB. The file type is 'Text Document'.