

Introducción

Departamento de Automática



Universidad de Alcalá

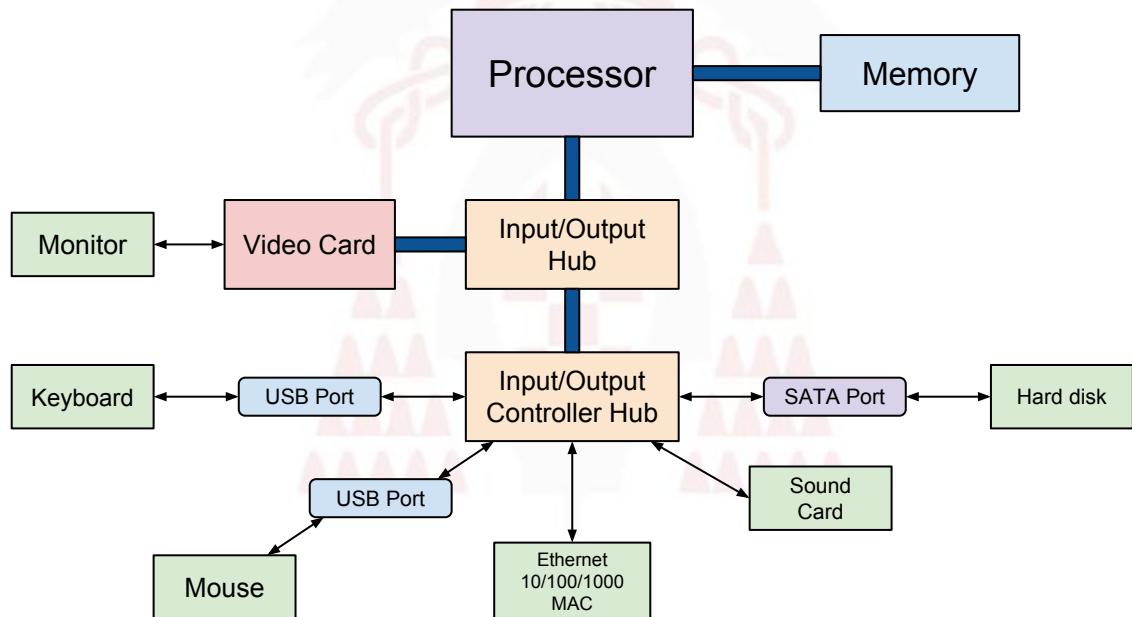


/gso>

Índice

- **Introducción a la asignatura**
 - Una nueva máquina
 - ¿Cómo usar esa máquina?
 - El Sistema Operativo
- **Los sistemas operativos dentro de la Ingeniería de Computadores**
 - Esquema general
 - Arquitectura de Computadores: Conceptos básicos
- **Evolución histórica de los SSOO**
 - Modelo de máquina desnuda
 - Monitor simple residente
 - Sistemas de procesamiento por lotes
 - Multiprogramación
 - Sistemas de tiempo compartido
 - Sistemas de tiempo real

¿Qué aporta esta máquina?



Problemas para utilizar la máquina

- Cada elemento *hardware* es diferente y su uso es complicado
 - ¿Tengo que aprender a usar cada uno de los dispositivos?
 - ¿Y si el fabricante cambia el modelo? ¿Tendré que volver a aprender cómo se usa?
- Varios usuarios/programas pueden compartir el mismo *hardware*
 - Recursos limitados, ¿quién los repartirá?
 - ¿Y si dos o más participantes quieren el mismo recurso?
 - ¿Cómo se asegura la privacidad?
 - ¿Los demás pueden ver y alterar mis datos?
- ¿Podrá utilizar la máquina alguien que no sepa lo que es el *hardware* y el *software*?

Introducción a la asignatura

Los sistemas operativos dentro de la Ingeniería de Computadores
Evolución histórica de los SSOO

¿Qué quiere el usuario?

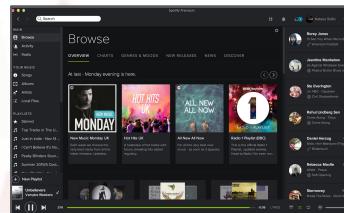
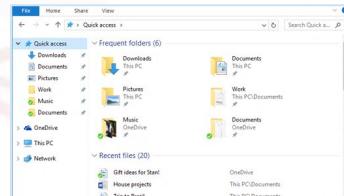
```
/home/mars$ pwd
/home/mars
mars@marsmain ~ $ cd /usr/portage/app-shells/bash
mars@marsmain /usr/portage/app-shells/bash % ls -al
total 130
drwxr-xr-x  3 portage portage 1024 Jul 25 10:06 .
drwxr-xr-x 33 portage portage 1024 Aug  7 22:39 ..
-rw-r--r--  1 root   root    35886 Jul 25 10:06 ChangeLog
-rw-r--r--  1 root   root    27005 Jul 25 10:06 Manifest
-rw-r--r--  1 portage portage  556 Mar 23 21:37 bash-3.2-r1.ebuild
-rw-r--r--  1 portage portage 5977 Mar 23 21:37 bash-3.2.p39.ebuild
-rw-r--r--  1 portage portage 6151 Apr  5 14:37 bash-3.2.p48-r1.ebuild
-rw-r--r--  1 portage portage 5988 Mar 23 21:37 bash-3.2.p48.ebuild
-rw-r--r--  1 portage portage 5643 Apr  5 14:37 bash-4.0.p10-r1.ebuild
-rw-r--r--  1 portage portage 6230 Apr  5 14:37 bash-4.0.p10.ebuild
-rw-r--r--  1 portage portage 5649 Apr 14 05:52 bash-4.0.p17-r1.ebuild
-rw-r--r--  1 portage portage 5532 Apr 14 05:52 bash-4.0.p17.ebuild
-rw-r--r--  1 portage portage 5660 May 30 03:35 bash-4.0.p24.ebuild
-rw-r--r--  1 portage portage 5660 May 30 03:35 bash-4.0.p24.ebuild
drwxr-xr-x  2 portage portage 2048 May 30 03:35 files
-rw-r--r--  1 portage portage 468 Feb  9 04:35 metadata.xml
mars@marsmain /usr/portage/app-shells/bash % cat metadata.xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE pkgmetadata SYSTEM "http://www.gentoo.org/dtd/metadata.dtd">
<pkgmetadata>
  <herd>base-system</herd>
  <use>
    <flag name='bashlogger'>Log ALL commands typed into bash; should ONLY be
      used in restricted environments such as honeypots</flag>
    <flag name='net'>Enable /dev/tcp/host/port redirection</flag>
    <flag name='plugins'>Add support for loading builtins at runtime via
      'enable'</flag>
  </use>
</pkgmetadata>
```

Una nueva máquina

¿Cómo utilizar la máquina de la mejor forma posible?
El Sistema Operativo



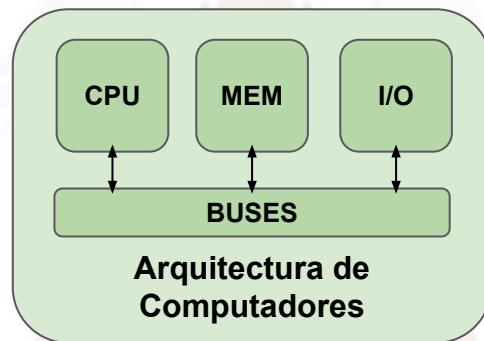
Mediante abstracciones



Algunas definiciones

- S. Sánchez: conjunto de programas que, por medio de abstracciones, ponen el *hardware* del ordenador, de modo seguro, a disposición del usuario
- H. Deitel: programa que actúa como interfaz entre el usuario de un ordenador y el hardware del mismo, ofreciendo el entorno necesario para que el usuario pueda ejecutar programas
- H. Katzan: conjunto de programas y datos que ayudan a crear otros programas y a controlar su ejecución
- S. Madnik y J. Donovan: conjunto de programas que gestionan los recursos del sistema, optimizan su uso y resuelven conflictos

Esquema general



Compiladores



CLang/LLVM



GCC



Visual Studio

Sistemas Operativos



macOS



Windows

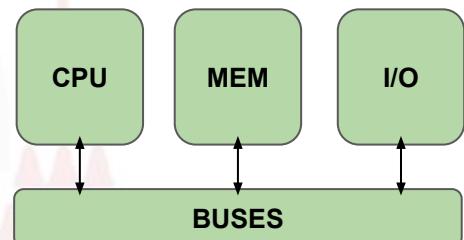
Definición y elementos fundamentales

Arquitectura

Se refiere a los atributos de un sistema que son visibles a un programador o, dicho de otra manera, aquellos que tienen un impacto directo en la ejecución de un programa

Elementos fundamentales

- Memoria
- CPU
- Buses
- Entrada y salida



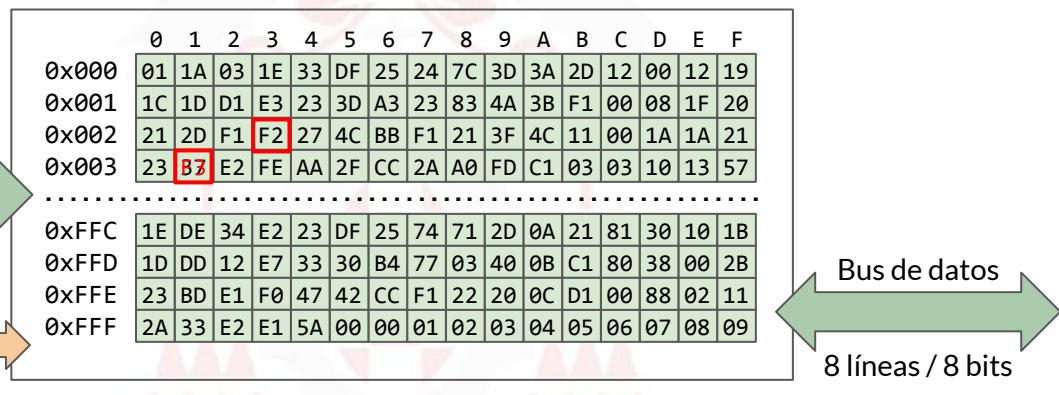
Memoria (1/3)

Características

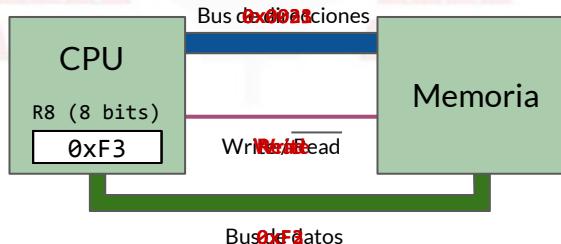
- Permite el almacenamiento de datos y programas en un computador
- Es una amplia tabla en la que cada dato se almacena en una posición
- El número de posiciones de la tabla determina el tamaño de la memoria
- El tamaño del dato almacenado y el número de posiciones dependen del tipo de diseño
- Los programas deben estar cargados en memoria principal para ser ejecutados

Memoria (2/3)

Memoria de 65536 posiciones de 8 bits

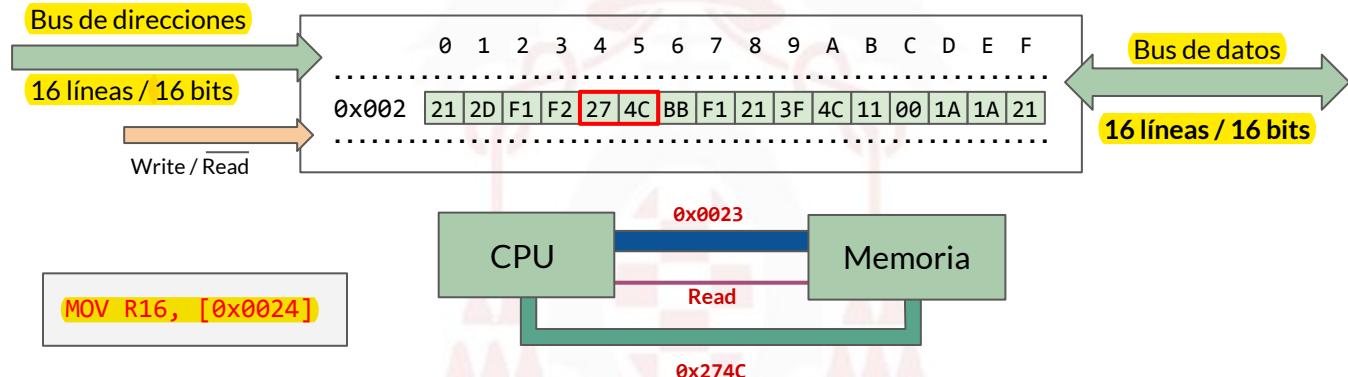


MOV B R8, [0x0023]
INC B R8
MOV [0x0031], R8



Memoria (3/3)

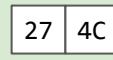
Memoria de 65536 posiciones de 8 bits



Big-endian

Memoria

0x0024 | 0x0025



MSB | LSB

R16 (16 bits)

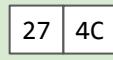
0x274C

MSB - el bit más significativo

Little-endian

Memoria

0x0024 | 0x0025



LSB | MSB

R16 (16 bits)

0x4C27

Unidad Central de Procesos (CPU)

Concepto

CPU

- CPU = procesador = microprocesador (en un chip)
- Consta de:
 - ALU (Unidad Aritmético-Lógica): realiza funciones aritméticas (suma, resta, etc.) y lógicas a nivel de bit (AND, OR, etc.)
 - Unidad de Control: interpreta las instrucciones
- Controla el resto de componentes del sistema y se encarga de ejecutar las instrucciones de los programas

Unidad Central de Procesos (CPU)

Modelo de programación

El modelo de programación incluye todos los elementos que la CPU pone a disposición de los programadores

- Elementos de almacenamiento: registros generales, contador de programa, registro de estado, puntero de pila, mapa de memoria y mapa de entrada/salida
- Juego de instrucciones y modos de direccionamiento

Unidad Central de Procesos (CPU)

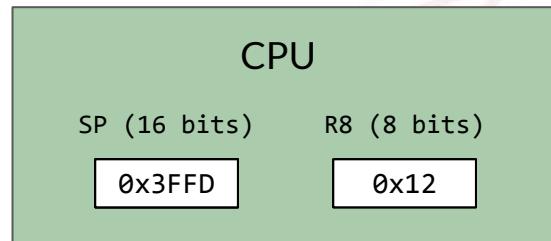
Registros de la CPU

Los registros son los elementos de almacenamiento que existen dentro de la CPU

- Registros de propósito general
- Registros de datos
- Registros de direcciones
 - Puntero de pila (*Stack Pointer o SP*): registro que apunta a la cabecera de la pila (operaciones PUSH y POP)
- Registros de control y de estado

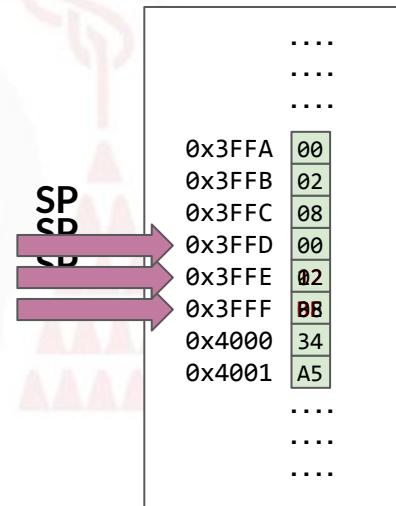
Unidad Central de Procesos (CPU)

Registro puntero de pila



PUSHB R8
PUSHB 0x12
POPB R8

Memoria de 65536 bytes



Unidad Central de Procesos (CPU)

Registros de control

Los **registros de control** se emplean para determinar el funcionamiento de la CPU:

- **Contador de programa (Instruction Pointer o IP):** contiene la dirección de la siguiente instrucción a ejecutar.
- **Palabra de estado del programa (PSW):** registro(s) con flags (acarreo, desbordamiento, etc.) fijados por el hardware de la CPU.

Modelo de programación

Juego de instrucciones

- Una **instrucción máquina** es una **secuencia** de **bits** que **representa** una **operación** a **realizar** y **dónde** se **encuentran** los **operandos** **necesarios**. Cada **instrucción** tiene **asignado** un **código máquina**
- A los **códigos máquina** se le **asigna** un **mnémónico**, más **cercano** al **lenguaje natural**
- Las **instrucciones** pueden **necesar** un **número variable** de **operandos**
- **No todo el mundo** puede **ejecutar** todas las **instrucciones**
- El **juego de instrucciones define** qué **operaciones** puede **realizar** la **CPU** (¿puede la CPU multiplicar matrices de números?)

Modelo de programación

Tipos de instrucciones

Aritméticas

ADD	SUB	MUL
sumar	restar	multiplicar

Lógicas bit-a-bit

AND	OR	NOT
Y-lógica	O-lógica	negación

Transferencia de datos

MOV	PUSH	POP
mover	apilar	extraer

RISC,
CISC

Control/Salto

CALL	RET
llamar a rutina	retorno de rutina
JMP	JNE
salto incondicional	salto si no son iguales

Modelo de programación

Modos de direccionamiento

Modos de direccionamiento

Formas de especificar e interpretar los operandos de una instrucción

Tipos de direccionamiento

- **Implícito:** el propio código de operación indica sobre qué operando actúa la operación
- **Explícito**
 - **Inmediato:** el operando se incluye en la propia instrucción
 - **Directo:** el operando se referencia con su dirección en memoria ADD A,[0x0100]
 - Otros: **indirecto, por registro, etc.** ADD A,[B]

Unidad Central de Procesos (CPU)

Ciclo de ejecución de una instrucción

Ciclo de ejecución

- Búsqueda de la instrucción máquina
- Interpretar la instrucción leída
- Leer los datos de memoria referenciados en la instrucción
- Ejecutar la instrucción
- Almacenar los resultados de la ejecución

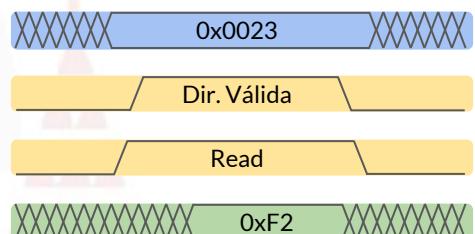
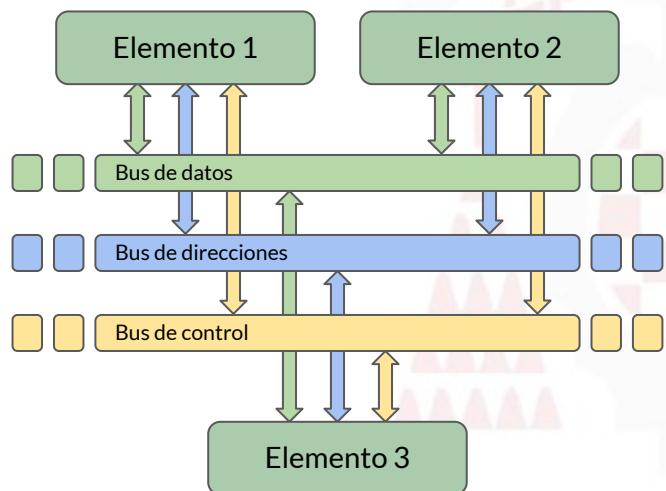
Buses

Descripción general

- Son los encargados de conectar entre sí los distintos componentes del sistema
- Inicialmente eran pasivos (cables), pero actualmente son elementos muy complejos
- Un bus se define por varios parámetros:
 - Sus conectores y características físicas
 - Su topología (serie, paralelo, estrella, etc.)
 - Sus señales y protocolos

Buses

Descripción general



Entrada y salida

Objetivo y características generales

Objetivo: Comunicación con el mundo exterior para ...

- Obtener los programas que hay que ejecutar (y colocarlos en memoria)
- Obtener los datos que hay que procesar
- Comunicar con usuarios humanos y con otras máquinas

Características

- La naturaleza de cada dispositivo es totalmente diferente
- Todos deben comunicar con la CPU (buses)
- Es necesario establecer una arquitectura común: parte específica y parte común (controlador)

Entrada y salida

Arquitectura de un dispositivo de E/S: **Controladores**

- Son **elementos hardware**
- Se **conectan** a los **buses** del **computador** y **ofrecen** **varios registros**:
 - Registro de **datos**
 - Registro de **control**
 - Registro de **estado**
- Algunos dispositivos complejos utilizan otros elementos (e.g. **framebuffer**)

Entrada y salida

Técnicas de E/S

- Los dispositivo necesitan atención (¿urgente?)
- Dos aproximaciones:
 - Que la CPU le pregunte al dispositivo (*polling*)
 - ¿Con qué frecuencia?
 - Carga de trabajo adicional para la CPU
 - Ocupa los buses y los dispositivos de forma innecesaria
 - Sencillo, no necesita *hardware* adicional
 - Que el dispositivo interrumpa a la CPU cuando lo necesite
 - Necesita *hardware* adicional
 - Necesita que la CPU disponga de mecanismo de interrupciones

Entrada y salida

Mecanismo de interrupciones

1. La CPU ejecuta un programa normalmente
2. Un dispositivo requiere atención y solicita una interrupción a la CPU
3. La CPU deja de ejecutar el programa y pasa a ejecutar otro programa denominado ISR (*Interrupt Service Routine*)
4. La ISR atiende al dispositivo y da por finalizada la interrupción
5. La CPU reanuda la ejecución del programa en el punto en el que fue interrumpida

Modelo de máquina desnuda

Objetivo

Ejecutar programas almacenados en memoria

- Se programaba directamente sobre el *hardware*
- No existía nada similar al sistema operativo
- Los usuarios introducían código máquina mediante interruptores
- Problemas:
 - El usuario debía conocer todo el *hardware*
 - Un cambio mínimo en el *hardware* invalidaba todo el programa
 - Se repite entre programas

Monitor simple residente

Objetivo

Reutilizar código

- El código común se agrupa dentro de un monitor simple residente
 - El monitor es el primer germen de sistema operativo
 - Agrupa fundamentalmente rutinas de entrada/salida
 - Un monitor es, esencialmente, un manejador de dispositivo o driver
- El programador no tiene que programar directamente la E/S
- Se puede, por ejemplo, cambiar el lector de tarjetas sin cambiar los programas, basta con cambiar el monitor
- Problema: se pierde mucho tiempo entre la finalización de un proceso y el lanzamiento del siguiente

Sistemas de procesamiento por lotes (*sistemas batch*)

Objetivo

Reducir los tiempos de espera de E/S en la carga de programas

- Para la entrada/salida se utilizan ordenadores de bajo coste dedicados
- La ejecución de los programas se realiza por medio de ordenador de altas prestaciones y alto coste

Sistemas de procesamiento por lotes (sistemas *batch*)

Modo de operación

- Un operario introduce las tarjetas perforadas en un ordenador de bajo coste
 - Una vez se termina, se obtiene una cinta que es cargada en el ordenador principal
 - El ordenador principal va ejecutando los programas de la cinta mientras el operario introduce nuevos programas
 - Cuando el ordenador principal termina, el operario retira la cinta y la introduce en un tercer ordenador que imprime los resultados
- Problema: hace falta la intervención de un operario humano.

Multiprogramación

Objetivo y requerimientos *hardware*

Objetivo

Solapar operaciones de E/S con la utilización de la CPU en la misma máquina

- Supone un salto de gran importancia y dificultad: tener más de un programa cargado en memoria a la vez
- Mientras se realiza una operación de E/S el microprocesador ejecuta otro programa de los que están en memoria
- Surgen nuevas necesidades *hardware*:
 - Interrupciones
 - Acceso directo a memoria (DMA)

Multiprogramación

Requerimientos del sistema operativo

- Surgen la mayoría de los problemas clásicos de los SSOO:
 - Gestión de la memoria
 - Planificación del procesador
 - Planificación de los dispositivos
 - Protección de la ejecución de diferentes programas
 - Control de la concurrencia
- Problemas:
 - Un programa en ejecución puede monopolizar CPU
 - Poco adecuado para sistemas interactivos

Sistemas de tiempo compartido

Objetivo

Garantizar a los procesos el acceso equitativo a la CPU

- Protección contra monopolización de la CPU; interrupciones
- El tiempo de ejecución asignado a cada programa se divide en quantum
- Si un programa en ejecución agota su quantum, la CPU pasa a ejecutar otro programa (round-robin)
- Los SSOO de tiempo compartido son los habituales actualmente en la informática de consumo

Sistemas de tiempo real

Concepto y características

Objetivo

Garantizar que los programas se ejecutan dentro de un plazo de tiempo acotado

- Para satisfacer restricciones de tiempo real no basta con tener un hardware más potente
- Tiempo real no es sinónimo de rápido
- Suelen encontrarse en sistemas empotrados
- Dos tipos de aplicaciones de tiempo real:
 - Duras
 - Blandas

Referencias bibliográficas

- Sebastián Sánchez. *Sistemas Operativos*. Segunda edición. Universidad de Alcalá - Servicio de Publicaciones, 2005
- A. S. Tanenbaum. *Sistemas Operativos Modernos*. Tercera edición. Prentice Hall, 2009
- William Stallings. *Computer organization and architecture*. Décima edición. Pearson, 2015

La consola de Linux

Departamento de Automática



Universidad de Alcalá



/gso>

Índice

- [El intérprete de órdenes](#)
 - Conseguir ayuda
- [El sistema de archivos](#)
 - Navegación por el sistema de archivos
 - Manipulación de archivos y directorios
 - Permisos
- [Trabajando con comandos](#)
 - Control de trabajos
- [Entrada/salida](#)
 - Redirecciones
 - Tuberías
- [Ejercicios propuestos](#)

El intérprete de órdenes El sistema de archivos Trabajando con comandos Entrada/salida Ejercicios propuestos	Descripción general Conseguir ayuda
<h3>Descripción general</h3> <ul style="list-style-type: none">● Es un programa que permite al usuario lanzar comandos para que los ejecute el sistema operativo● Ejemplos de comandos:<ul style="list-style-type: none">○ date: muestra o fija la fecha y hora del sistema○ cal: muestra un calendario○ df: muestra la cantidad de espacio utilizado por el sistema de archivos○ exit: finaliza el intérprete de órdenes● Mantiene un histórico de los comandos que se han introducido● Permite auto-completar el nombre comandos (TAB) history	
Sistemas Operativos La consola de Linux 3 / 22	

Nosotros utilizamos el **intérprete** de órdenes **bash**.

Con el comando **df** se puede usar el modificador **-h** para obtener los valores en formato legible por humanos.

El histórico de comandos del intérprete nos permite una serie de funcionalidades:

- El comando **history** muestra la lista completa de comandos en pares (número de evento/comando, comando)
- Si pulsamos la **flecha-arriba** podemos ir **navegando** por los **últimos comandos** que se han introducido
- Si pulsamos **Ctrl+R** podemos **buscar dentro** del **histórico** de **comandos**
- Anteponiendo el carácter “!” antes de uno o más caracteres, e.g. !cal, podemos **reproducir** el **último comando** lanzado que **comienza** por **dichos caracteres**.

Pulsado la tecla **TAB** (tabulador), podemos **autocompletar** el **nombre** de **comandos**.

Conseguir ayuda

- La herramienta man permite **obtener ayuda** sobre distintos elementos del sistema:

```
$ man df  
$ man man
```

- Las **páginas de manual** están **divididas** en varias **secciones**:
 - 1 → Comandos generales
 - 2 → Llamadas al sistema
 - 3 → Biblioteca de funciones en C

Se **puede buscar** dentro de las **páginas del manual** pulsando el carácter **/** y a continuación la **cadena** de caracteres a buscar.

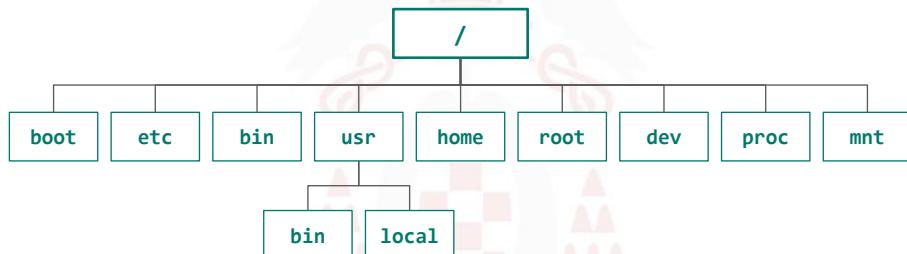
Se puede **buscar** un **comando** a **partir** del **texto** de su **descripción**:

```
$ man -k texto_a_buscar
```

Puede que una **misma entrada** en el manual **aparezca** en **más** de una **sección**, e.g. **read, write...** entonces **hay que especificar la sección concreta** en la que **queremos buscar**:

```
$ man 3 write
```

El árbol de directorios



- El comando `pwd` imprime el directorio de trabajo actual (*working directory*)
- El comando `ls` permite mostrar el contenido del directorio de trabajo actual

En Unix/Linux los archivos están organizados en lo que se conoce como **directorios**. Un directorio no es más que un archivo especial que contiene información que permite localizar otros archivos. Los directorios pueden contener a su vez nuevos directorios, que se denominan **subdirectorios**.

La jerarquía de directorios de Unix estandariza una serie de directorios con sus funciones asociadas:

- **/boot**: contiene el núcleo de Linux y los archivos de arranque del sistema
- **/etc**: contiene los archivos de configuración del sistema
- **/bin** y **/usr/bin**: contienen la mayoría de los programas del sistema. El directorio `/bin` almacena los programas esenciales, mientras que `/usr/bin` contiene las aplicaciones de usuario
- **/usr/local**: se utilizan para instalar programas para utilizarlos en la máquina de forma local. Estos programas en principio no pertenecen a la distribución de Linux, sino que son instalados a discreción por el propio usuario
- **/home**: contiene los directorios personales de los usuarios. Cada usuario registrado tiene un directorio dentro propio dentro de `/home`
- **/root**: es el directorio personal del usuario root (superusuario)
- **/dev**: directorio cuyos archivos representan a los dispositivos instalados en el sistema. En los sistemas Unix/Linux, se emplea el archivo como una abstracción de un dispositivo. De esta forma, cuando leemos o escribimos en uno de estos archivos, estamos realmente leyendo y escribiendo datos del dispositivo correspondiente.

- **/proc**: es un directorio virtual cuyos archivos nos permiten obtener información acerca del sistema y del funcionamiento del sistema operativo y de los procesos que están actualmente corriendo.
- **/mnt**: en algunos sistemas, como Ubuntu, se usa también otro directorio llamado **/media**. Este directorio se emplea normalmente como “punto de montaje” de sistemas de archivos distintos del principal. Por ejemplo, cuando introducimos una llave USB, el sistema “monta” los archivos de la llave colgando de este directorio.

Otros directorios:

- **/var**: contiene archivos que son modificados durante la ejecución del sistema como, por ejemplo, archivos de “log”.
- **/tmp**: directorio empleado por los programas para guardar archivos temporales. Se borra con cada reinicio del sistema.

El directorio de conexión (~), también llamado directorio **home**, es el directorio en el cual comienza un usuario cuando abre una sesión en Unix/Linux. Cada usuario tiene su propio directorio de conexión. Normalmente el directorio de conexión se ubica en **/home/usuario**, en donde **usuario** es el nombre del usuario.

Navegación por el sistema de archivos

Cambio del directorio actual de trabajo

- El comando `cd` permite cambiar el directorio actual de trabajo
- Para referenciar al directorio destino se pueden emplear rutas absolutas o relativas

Ejemplo de ruta absoluta:

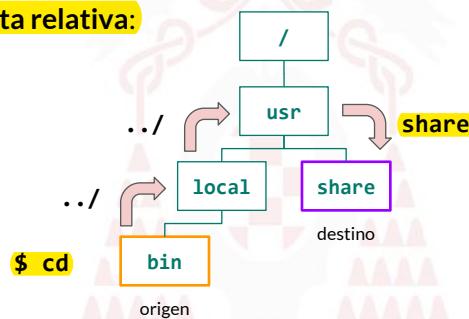


- Toma como origen el directorio raíz /
- Es independiente del directorio actual de trabajo

Navegación por el sistema de archivos

Cambio del directorio actual de trabajo

Ejemplo de ruta relativa:



- Toma como origen el directorio actual de trabajo
- Para navegar por el árbol se utiliza el directorio ..

Cada directorio tiene dos subdirectorios especiales:

- El directorio punto (.), que es una referencia al directorio actual de trabajo.
- El directorio punto-punto (..), que es una referencia al directorio padre del directorio actual de trabajo.

Navegación por el sistema de archivos

Listado de archivos empleando el comando `ls`

- El comando `ls` proporciona distintas opciones de visualización:
 - `-a`: muestra los **archivos ocultos**
 - `-l`: muestra un **listado detallado** de los **archivos**
- Ejemplo de listado detallado:

drwxr-xr-x	2	root	root	4096	ago	12	08:42	subversion
drwxrwxr-x	6	parraman	parraman	4096	mar	29	2017	local
-rw-rw-r--	1	parraman	parraman	462	nov	30	12:52	instr.txt

Diagrama que explica la estructura de los campos de la salida del comando ls:

- Nombre: Columna 9
- Última modificación: Columna 8
- Tamaño: Columna 5
- Grupo: Columna 4
- Dueño: Columna 3
- Número de enlaces: Columna 2
- Permisos del archivo: Columna 1

El primer carácter del conjunto “Permisos del archivo” informa sobre el **tipo** de archivo:

- **d**: directorio
- **-**: archivo regular
- **c**: dispositivo en modo carácter (e.g. teclado)
- **b**: dispositivo en modo bloque (e.g. el disco duro)

Los permisos generales se tratan en una transparencia posterior.

Manipulación de archivos y directorios

Inspección de archivos

- El comando `file` permite mostrar el tipo de un archivo
 - El tipo lo obtiene inspeccionando su “número mágico”
- El sistema proporciona distintos comandos para inspeccionar el contenido de archivos de “texto”:
 - `cat`: muestra el contenido completo de un archivo
 - `more`: muestra el contenido completo de un archivo de forma paginada
 - `less`: versión mejorada de `more` que permite navegar por las líneas del archivo

El número mágico de un archivo son unos bytes específicos que están almacenados dentro de un archivo en posiciones fijas (generalmente al comienzo del archivo) y que sirven para establecer su tipo.

Ejemplo de números mágicos:

- **JPEG**: los archivos comienzan con los bytes [0xFF, 0xD8] y terminan con los bytes [0xFF, 0xD9]
- **PDF**: los archivos comienzan con la cadena de caracteres "%PDF" (en hexadecimal: [0x25, 0x50, 0x44, 0x46])
- **ZIP**: los archivos comienzan los caracteres "PK" (en hexadecimal: [0x50, 0x4B])

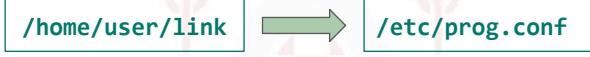
Para comprobar el número mágico se puede usar el comando `hexdump`:

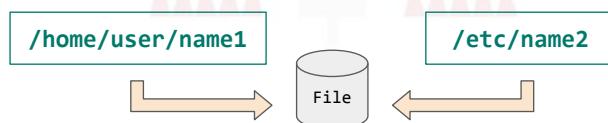
```
$ hexdump -C archivo | less
```

Ejemplo de archivos interesantes para mostrar: `/proc/cpuinfo`, `/proc/meminfo`, `/etc/passwd`

Manipulación de archivos y directorios

Enlaces simbólicos y fuertes

- Un enlace simbólico (*soft link*) es un archivo que contiene una referencia a otro archivo:

- Un enlace fuerte (*hard link*) es una entrada en un directorio asociada a un archivo
 - Un mismo archivo puede tener más de una entrada en el árbol de directorios



Los enlaces, tanto los simbólicos como los duros, se crean utilizando el comando `ln`.

Para crear un enlace simbólico:

```
$ ln -s origen destino
```

Para crear un enlace fuerte:

```
$ ln origen destino
```

Después de crear un enlace simbólico, se puede mostrar el “contenido” del enlace haciendo un listado detallado del archivo (con `ls -l`). Si se borra el archivo destino, el enlace se “rompe”.

Después de crear un enlace fuerte, el listado detallado mostrará cómo el número de enlaces del archivo ha aumentado. Con la orden `ls -li` se puede comprobar que el número de nodo-i es el mismo para los dos enlaces fuertes. Si se borra uno de los dos archivos, el otro sigue estando. No se borra el archivo completamente hasta que no se borran todos sus enlaces.

Manipulación de archivos y directorios

Comodines (*wildcards*) y variables

- El intérprete define comodines (*wildcards*) que son sustituidos en las órdenes antes de llamar al programa:
 - * (asterisco): representa un conjunto cualquiera de caracteres
 - ?: representa un único carácter cualquiera
- Se pueden definir variables que también son sustituidas en la orden:
 - Variables *shell*: variables propias de la instancia actual
 - Variables de entorno: variables que son heredadas por los programas lanzados desde el intérprete

Para definir una variable shell:

```
$ NOMBRE=valor
```

Para definir una variable de entorno:

```
$ export NOMBRE=valor
```

Las variables se pueden introducir como parte de la orden anteponiendo el carácter '\$' al nombre de la variable, e.g.:

```
$ echo $PATH
```

Antes de llamar al comando echo, el intérprete de órdenes sustituye PATH por el valor de la variable.

Las variables de entorno definidas se pueden mostrar con el comando env.

Además de variables, el intérprete de órdenes puede sustituir el resultado de la ejecución de una orden:

```
$ DIRECTORIO=$(pwd)  
$ echo $DIRECTORIO
```

Manipulación de archivos y directorios

Operaciones generales

- Existen distintos comandos que permiten realizar operaciones generales sobre los archivos:
 - **cp:** copia un archivo en otro
 - **mkdir:** crea un nuevo directorio
 - **rm:** borra un archivo
 - **rmdir:** borra un directorio vacío
 - **mv:** mueve (o renombra) un archivo

Para borrar un directorio no vacío se puede emplear el comando **rm** con el modificador **-r** (recursivo).

Permisos de acceso

- Los permisos de acceso a los archivos se dividen en tres clases:
 - Permisos de lectura
 - Permisos de escritura
 - Permisos de ejecución
- Se definen tres conjuntos de permisos para cada archivo:
 - Permisos del propietario del archivo
 - Permisos del grupo propietario del archivo
 - Permisos del resto de usuarios
- Los permisos se pueden cambiar con la orden chmod

Los archivos tienen un usuario propietario y un grupo propietario.

Los permisos de los archivos se muestran en el listado detallado (ls -l)

La orden chmod permite cambiar los permisos del archivo. Los permisos se pueden expresar como un número de tres dígitos en octal. Cada uno de los dígitos representa los permisos de un conjunto (dueño, grupo y resto de usuarios). Cada uno de los tres dígitos de la representación binaria de uno de los números originales en octal establece una de las clases de permisos (lectura, escritura y ejecución, en este orden).

Trabajando con comandos

¿Qué es un comando?

- Existen distintos tipos de comandos:
 - Archivos ejecutables
 - Comandos internos del intérprete de órdenes
 - Scripts de comandos
 - Alias de comandos
- El comando `which` muestra la ruta de los archivos ejecutables

Ejemplo de comando interno: `cd`

Primera toma de contacto paso a paso con el `vim`: editar un archivo con dos comandos (e.g. un `echo` y un `ls`), darle permisos de ejecución con `chmod` y ejecutarlo.

El comando `alias` muestra todos los alias actualmente definidos en la sesión del intérprete de órdenes.

Trabajando con comandos

Control de trabajos

- Los comandos se pueden ejecutar en primer o en segundo plano
 - Para ejecutar un comando directamente en segundo plano hay que terminar la orden con el carácter &
- La orden jobs muestra todos los trabajos lanzados en segundo plano
- Cuando un programa se está ejecutando en primer plano, se puede detener pulsando Ctrl+Z
- La orden fg pasa un trabajo detenido a primer plano
- La orden bg pasa un trabajo detenido a segundo plano

Prueba de ejecución de un comando en primer plano y un comando en segundo plano:

```
$ sleep 10  
$ sleep 10 &
```

Ejemplo con xload: http://linuxcommand.org/lc3_tts0100.php. Se pueden lanzar, detener con Ctrl + Z, pasar a primer plano o a segundo plano con fg o bg.

Trabajando con comandos

Monitorización de procesos

- Existen órdenes que proporcionan información acerca de los procesos del sistema:
 - **ps**: lista los procesos en curso y muestra información sobre ellos.
 - **top**: muestra información interactiva sobre los procesos en curso
- El sistema proporciona un mecanismo de comunicación asíncrono entre procesos: las señales
 - **kill**: permite enviar una señal a un proceso

Todos los procesos en curso tienen un identificador único (*Process Identifier* o PID).

La orden kill se utiliza de la siguiente forma:

```
$ kill -N PID
```

Donde N es el número de señal. Algunos ejemplos de señales:

- 2 → **SIGINT**: Interrupción. Se genera al pulsar Ctrl+C.
- 9 → **SIGKILL**: Muerte segura. No puede ser ignorada, se utiliza para matar procesos que se niegan a “suicidarse”.
- 15 → **SIGTERM**: Finalización de proceso (software). Es similar a la 9, pero puede ser ignorada.

Descripción general



- Todos los procesos tienen asignados tres archivos especiales:
 - 0 → entrada estándar (por defecto: teclado del terminal)
 - 1 → salida estándar (por defecto: pantalla del terminal)
 - 2 → salida estándar de error (por defecto: pantalla del terminal)
- Los procesos se comunican con el exterior leyendo/escribiendo de/en esos archivos

Cuando un proceso necesita leer datos, por lo general lo hace a través de la entrada estándar, esto es, leyendo del archivo con descriptor 0 que, por defecto, hace referencia al teclado. De la misma manera, cuando un proceso necesita mostrar información, generalmente lo hará escribiendo en el archivo con descriptor 1, correspondiente a la salida estándar. Si la información a mostrar se corresponde con algún error de ejecución, el proceso puede utilizar la salida estándar de error, esto es, el archivo con descriptor 2.

Redirecciones

- Los archivos por defecto de la entrada y las salidas estándar se pueden modificar en la propia orden usando redirecciones:

- Redirección de la entrada estándar (<)

```
$ bc < operaciones
```

- Redirección de la salida estándar (>, >>)

```
$ echo $PATH > path
```

- Redirección de la salida estándar de error (2>, 2>>)

```
$ ls /foo 2> /dev/null
```

Tuberías

Descripción general

- Las **tuberías** (*pipes*) son un **mecanismo** de **conexión entre procesos**
- Permiten conectar la **salida** **estándar** de un **proceso** con la **entrada** **estándar** de **otro**, por ejemplo:

```
$ ls | wc -w
```

- Pueden conectar dos o más procesos:

```
$ orden_1 | orden_2 | orden_3 | ...
```

El intérprete de órdenes
El sistema de archivos
Trabajando con comandos
Entrada/salida
Ejercicios propuestos

Descripción general
Redirecciones
Tuberías

Tuberías

Filtros

- Un filtro es un programa que lee de su entrada estándar y escribe en su salida estándar
- Normalmente los filtros pueden aceptar uno o varios archivos como parámetros (no es necesario redireccionar la entrada)
- Existen numerosos filtros en el sistema:
 - sort: ordena las líneas de entrada
 - cut: extrae uno o varios campos en las líneas de entrada
 - grep: busca patrones de texto en las líneas de entrada
 - tail: muestra las N últimas líneas de entrada
 - head: muestra las N primeras líneas de entrada

Sistemas Operativos | La consola de Linux | 20 / 22

El comando **cut** permite extraer caracteres concretos de las líneas de un archivo. Por ejemplo:

```
$ cut -c 1-9 archivo
```

extrae los nueve primeros caracteres de todas las líneas del archivo y los vuela por la pantalla.

También permite extraer campos empleando delimitadores, por ejemplo:

```
$ cut -f 3 -d ':' /etc/passwd
```

Extrae el tercer campo separado por el carácter ':' del archivo /etc/passwd.

Ejercicios propuestos

1. Mostrar la fecha actual con el formato 31-12-2018 (día, mes y año separados por un guión) `date +%m-%d-%Y`
2. Copiar el archivo /etc/passwd al directorio actual de trabajo:
 - o Empleando rutas absolutas `cp /etc/passwd /home/lenguajes/Escritorio/dani`
 - o Empleando rutas relativas `cp ../../etc/passwd .`
3. Salvar el espacio utilizado en el sistema en un archivo con el nombre: `diskreport-{month}-{day}-{hour}-{min}-{sec}`
4. ¿Cuántas imágenes hay en la página web <https://www.uah.es>?
 - o Buscar el comando curl `wget -q https://www.uah.es`
 - o Filtrar con grep las líneas que contengan la etiqueta img
 - o Contar el número de líneas
`wget -q http://www.uah.es | grep -Eo "img.*(|>)" index.html | wc -l`

3. `df -h / | awk 'END { print $3 }' | cat > diskreport-$(date '+ %m-%d-%H-%M-%S')`

Si el comando curl no estuviese disponible, se puede utilizar el comando wget en su lugar:

```
$ wget -qO - https://www.uah.es
```

Referencias bibliográficas

- Sebastián Sánchez Prieto y Oscar García Población. *Unix y Linux, guía práctica*. Ra-Ma, 3 Ed., 2004.
- The Linux Command Line. Disponible en:
http://linuxcommand.org/lc3_learning_the_shell.php

Introducción al simulador

Departamento de Automática



Universidad de Alcalá



/gs0>

Índice

- [Introducción al simulador](#)
- [Mapa de memoria](#)
 - [Proyecciones de dispositivos de entrada/salida](#)
- [Procesador](#)
 - [Registros](#)
 - [Instrucciones](#)
- [Ciclo de ejecución](#)
- [Ejercicios propuestos](#)

Descripción general

Sitio web

<https://parraman.github.io/asm-simulator/>

Proyecto github (código fuente)

<https://github.com/parraman/asm-simulator>

- Simulador de una arquitectura de 16 bits
 - Angular 2+ (TypeScript)
 - Bootstrap 3
 - CodeMirror

Introducción al simulador
 Mapa de memoria
 Procesador
 Ciclo de ejecución
 Ejercicios propuestos

Descripción general
 Arquitectura del simulador

Descripción general

Editor de código

```
Code (Sample 1, Sample 2, Sample 3, Sample 4, Sample 5)
1    .org 1000
2    .string "Hello World!" ; Write the textual display
3    .text
4    .start
5    hello: .db 'Hello World!', 0 ; Output string
6    .end
7    .model small
8    .stack 100h
9    start:
10   mov ax, 0001h
11   mov bx, 0000h
12   mov cx, 0000h
13   mov dx, 0000h
14   mov si, offset hello
15   mov di, offset hello
16   mov ah, 09h
17   int 21h
18   add di, 2
19   cmp di, si
20   jne .loop
21   mov ah, 4ch
22   int 21h
23   .loop:
24   mov ah, 01h
25   int 21h
26   cmp al, 0dh
27   jne .loop
28   mov ah, 09h
29   int 21h
30   add di, 2
31   cmp di, si
32   jne .loop
33   mov ah, 4ch
34   int 21h
35   .exit:
```

Dispositivos E/S

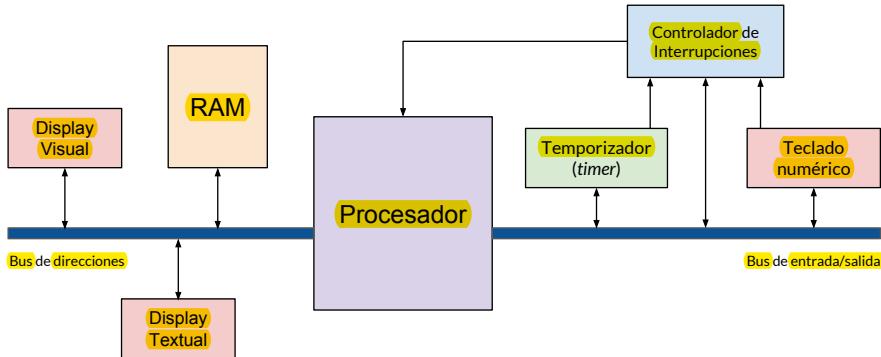
CPU (Registros)

Mapa de memoria

Mapa de E/S

Sistemas Operativos
Introducción al simulador
4 / 17

Arquitectura del simulador



Descripción básica de los componentes básicos del sistema simulado:

- **Procesador**
- **Memoria**: espacio de direcciones de 16 bits. Solo se usan los primeros 1024 bytes
- **Controlador de interrupciones**: centraliza las distintas fuentes de interrupción. Se explica en la siguiente práctica
- **Timer**: permite programar interrupciones (alarma). Se explica en la siguiente práctica
- **Dispositivos de E/S**: display visual, display textual y teclado numérico

Mapa de memoria

Descripción general

- Mapa de memoria de 1024 bytes

- Permite la edición manual de celdas:

(Ctrl + click)
ó
(Cmd + click)

- Los valores de las celdas tienen formato hexadecimal

Memory																Split
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
000	2E	00	10	48	65	6C	6C	6F	20	57	6F	72	6C	64	21	00
001	86	04	00	FF	06	02	00	03	06	03	02	F0	46	00	20	00
002	3B	00	3B	01	06	01	00	00	0B	0A	00	02	0D	00	03	0A
003	22	0E	22	10	2A	0C	00	02	36	00	28	43	01	43	00	47
004	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
005	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
006	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
007	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
008	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
009	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00B	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00C	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00D	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00E	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00F	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

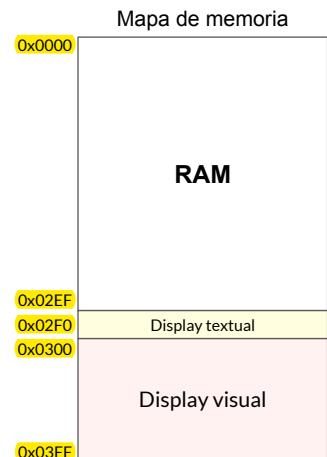
Ejemplos: edición manual de una o varias celdas

Mapa de memoria

Proyecciones de dispositivos de E/S

El mapa de memoria está dividido en tres zonas:

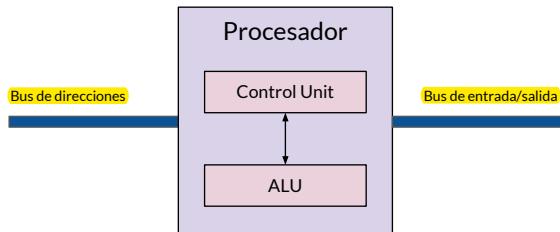
- Zona de RAM: memoria de lectura/escritura donde almacenar datos y código
- Zona del display textual: cada celda se corresponde con una letra. El valor será el carácter ASCII a mostrar
- Zona del display visual: cada celda se corresponde con un pixel. El valor será el color del pixel



Ejemplos: edición manual de una o varias celdas de las proyecciones del display textual y del display visual

Procesador (CPU)

Descripción general



El procesador consta de:

- **Unidad de control (Control Unit):** se encarga de obtener de memoria, interpretar y ejecutar las instrucciones
- **Unidad Aritmético-Lógica (ALU):** realiza funciones aritméticas y lógicas a nivel de bit

Procesador (CPU)

Registros

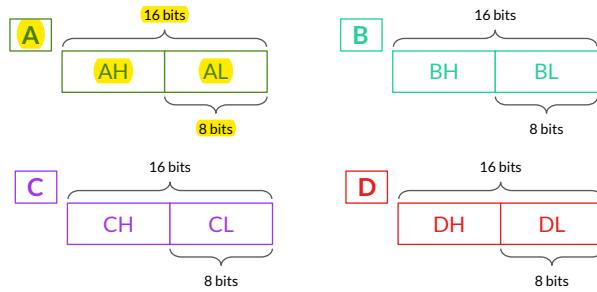
CPU Registers							
A 0000	B 0000	C 0000	D 0000				
IP 0010	SSP 0000	SR S -----		M	C	Z	F H
USP 0000			1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0				

El procesador incluye tres tipos de registros:

- Registros de propósito general
- Registros de direcciones
- Registro de Estado (*Status Register*)

Procesador (CPU)

Registros de propósito general

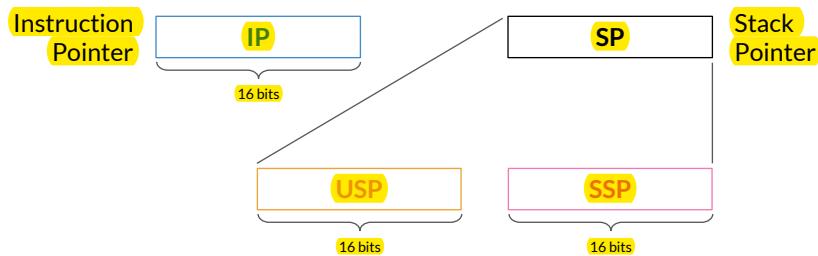


El procesador incluye cuatro registros de propósito general:

- Tienen un tamaño de 16 bits
- Pueden ser accedidos en modo palabra o modo byte

Procesador (CPU)

Registros de direcciones



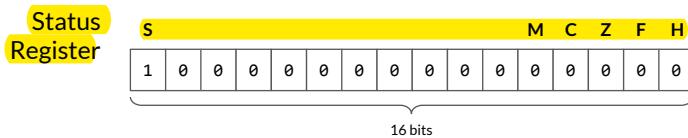
El procesador mantiene dos punteros de pila:

- Pila de modo supervisor (SSP)
- Pila de modo usuario (USP)

La selección de pila se hace de forma automática. Si la CPU está en modo supervisor, el registro SP es el SSP. Si está en modo usuario, el registro es el USP. Todas las instrucciones que direccionan el registro de pila de forma explícita, siempre se refieren al SP genérico. Ese SP es, por tanto, como un “alias” del verdadero registro puntero de pila.

Procesador (CPU)

Registro de estado



El registro de estado define los siguientes flags:

- **S: Supervisor Mode**
- **M: Interrupt Mask**
- **C: Carry flag**
- **Z: Zero flag**
- **F: Fault mode**
- **H: Halt mode**

- **Supervisor Mode Flag.** Activado si la CPU está en modo supervisor y desactivado si está en modo usuario. La CPU arranca en modo supervisor. Los modos de operación se explican más en detalle prácticas posteriores.
- **Interrupt Mask Flag.** Activado si la CPU permite interrupciones y desactivado si las interrupciones están deshabilitadas. Se explica más en detalle en prácticas posteriores.
- **Carry Flag.** Se activa si la última operación aritmético-lógica ha producido acarreo. Por ejemplo, si la suma de dos enteros produce un desbordamiento o si la resta de dos enteros produce un número negativo.
- **Zero flag.** Se activa si la última operación aritmético-lógica ha dado un resultado de cero. Si el resultado último fue distinto de cero, el bit estará desactivado.
- **Fault mode flag.** Activado si la CPU se encuentra en modo fallo (error irrecuperable que requiere un reinicio).
- **Halt mode flag.** Activado si la CPU se encuentra en modo bloqueo. En este modo, la CPU sigue activada pero deja de ejecutar instrucciones hasta que se produce una interrupción. Se explica más en detalle en prácticas posteriores.

Juego de instrucciones

Formato de instrucción



- Una instrucción está formada por:
 - Un código de operación (opcode) de 8 bits
 - Niguno, uno o dos operandos de 8 o 16 bits
- El código de operación determina el tipo de instrucción a ejecutar y los operandos que puedan necesitarse
- Cada instrucción está identificada por un mnemónico (e.g. MOV, ADD, etc)

Un mismo mnemónico (e.g. MOV) se puede utilizar para identificar una o varias instrucciones dependiendo de los distintos tipos de operandos. En ese caso el procesador define un opcode distinto para cada uno de los distintos conjuntos de operandos.

Juego de instrucciones

Tipos de operandos

Nombre	Descripción	Tamaño
BYTE	Valor inmediato de 8 bits	1 byte
WORD	Valor inmediato de 16 bits	2 bytes
ADDRESS	Dirección de 16 bits	2 bytes
REGISTER_8BITS	Registro de 8 bits	1 byte
REGISTER_16BITS	Registro de 16 bits	1 byte
REGADDRESS	Direccionamiento por registro + offset	2 bytes

REGISTER_16BITS: este operando codifica el número de referencia o índice de uno de los registros de 16 bits que implementa la CPU. Los índices de cada registro son los siguientes:

- A - Registro de propósito general A: 0 (0x00)
- B - Registro de propósito general B: 1 (0x01)
- C - Registro de propósito general C: 2 (0x02)
- D - Registro de propósito general D: 3 (0x03)
- SP - Registro puntero de pila: 4 (0x04)

REGISTER_8BITS: este operando codifica el número de referencia o índice de uno de los registros de 8 bits que implementa la CPU. Los índices de cada registro son los siguientes:

- AH - MSB del Registro A: 9 (0x09)
- AL - LSB del Registro A: 10 (0x0A)
- BH - MSB del Registro B: 11 (0x0B)
- BL - LSB del Registro B: 12 (0x0C)
- CH - MSB del Registro C: 13 (0x0D)
- CL - LSB del Registro C: 14 (0x0E)
- DH - MSB del Registro D: 15 (0x0F)
- DL - LSB del Registro D: 16 (0x10)

REGADDRESS: este operando codifica con 1 byte el número de referencia de un

registro de 16 bits y, con el otro byte, el offset a añadir al valor del parámetro para formar la dirección efectiva. El offset se codifica empleando complemento a dos [-128, 127].

Juego de instrucciones

Ejemplos

opcode	Mnem.	Operando 1	Operando 2	Descripción
6 (0x06)	MOV	REGISTER_16BITS	WORD	Mover un valor inmediato de 16 bits en un registro de 16 bits
33 (0x21)	INC	REGISTER_16BITS	-	Incrementar en una unidad el valor de un registro de 16 bits
5 (0x05)	MOV	ADDRESS	REGISTER_16BITS	Mover el valor de un registro de 16 bits a una dirección

Documentación sobre el juego de instrucciones

<http://asm-simulator.readthedocs.io/en/latest/instruction-set.html>

Codificar las instrucciones eligiendo un registro, un valor y una dirección de 16 bits.
Editar las celdas a partir de la 0x0000 de forma manual y ejecutar las instrucciones paso a paso.

Ciclo de ejecución de una instrucción

1. Carga del código de operación apuntado por el registro IP
2. Decodificación del código de operación
3. Carga de los operandos si los hubiere
4. Resolución de direccionamiento por registro si lo hubiere
5. Ejecución de la instrucción
6. Actualización del registro IP

➤ El log de eventos muestra de forma detallada los pasos del ciclo

Ejemplo con el log de eventos de la ejecución de las instrucciones de la transparencia anterior.

Comentar paso a paso el Ejemplo 1 (Sample 1) del simulador. El código de este ejemplo escribe en el display textual la cadena de caracteres “Hello world!”.

- Directivas de ensamblador (etiquetas y DB)
- Concepto de ensamblado (traducción a lenguaje máquina).
- Consecuencias del RESET (siempre comienza la ejecución por IP = 0)
- Ejecución paso a paso de CALL // RET. Ver que la dirección de retorno se almacena en la pila.
- Comprobar la modificación de los flags del registro de estado cuando se ejecuta la instrucción de comparación CMPB. Ver que la instrucción de salto condicional comprueba el valor del registro de estado para decidir si salta o no salta.

Ejercicios propuestos

Implementar un programa que imprima en el display textual la cadena “Hello world!” pero pasando a mayúsculas las ocurrencias de la letra “o”:

- El programa debe comprobar si la letra que se va a imprimir es una “o”. En caso afirmativo, debe pasarla a mayúscula antes de imprimirla
- La distancia en ASCII entre una letra mayúscula y su correspondiente minúscula es de 0x20 posiciones

En una primera aproximación al ejercicio, modificar la cadena original por una cadena en minúsculas (e.g. “hello world”) que se transforme a mayúsculas a medida que se imprimen los caracteres.

Proponer que revisen por su cuenta el código del Ejemplo 2 (Sample 2).

Interrupciones y entrada/salida

Departamento de Automática



Universidad de Alcalá



/gs0>

Índice

- [Mapa de entrada/salida](#)
 - Instrucciones IN/OUT
- [El temporizador periódico](#)
 - Modo de operación
- [El controlador de interrupciones](#)
 - Registros
 - Habilitación de interrupciones
- [Manejadores de interrupción](#)
 - Marco de interrupción
 - Procesamiento de una interrupción
- [Ejercicios propuestos](#)

Mapa de entrada/salida

Descripción general

- Mapa de registros de 16 bits de los dispositivos hardware
- Cada registro tiene una dirección fija de 16 bits
- El acceso al mapa de registros se realiza mediante instrucciones específicas

Input / Output Registers		
Address	Name	Value
0000	IRQMASK	0000
0001	IRQSTATUS	0000
0002	IRQE0I	0000
0003	TMRPRELOAD	0000
0004	TMRCOUNTER	0000
0005	KPDSTATUS	0000
0006	KPDDATA	0000



IN / OUT

Instrucciones IN/OUT

Instrucción IN

- Lee el valor de un registro de entrada/salida
- La dirección del registro se obtiene del primer y único operando
- El resultado se almacena siempre en el Registro A



- El primer operando puede ser:

opcode	Operando	Ejemplo
135 (0x87)	REGISTER_16BITS	IN B
136 (0x88)	REGADDRESS	IN [C+100]
137 (0x89)	ADDRESS	IN [0x0300]
138 (0x8A)	WORD	IN 0x2

Instrucciones IN/OUT

Instrucción OUT

- Escribe un valor en un registro de entrada/salida
- La dirección del registro se obtiene del primer y único operando
- El valor a escribir siempre se obtiene del Registro A

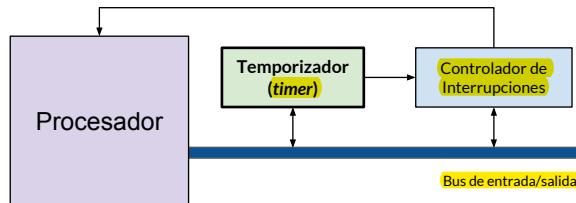
Instrucción
privilegiada

- El primer operando puede ser:

opcode	Operando	Ejemplo
139 (0x8B)	REGISTER_16BITS	OUT C
140 (0x8C)	REGADDRESS	OUT [B+100]
141 (0x8D)	ADDRESS	OUT [0x0200]
142 (0x8E)	WORD	OUT 0x1

El temporizador periódico

Descripción general



- Temporizador (timer) de 16 bits
- Permite generar interrupciones periódicas
- Implementa dos registros:
 - TMRPRELOAD: almacena el valor de pre-carga del contador
 - TMRCOUNTER: contiene el valor actual del contador
- Inicialmente el temporizador está desactivado

Mapa de entrada/salida El temporizador periódico El controlador de interrupciones Manejadores de interrupción Ejercicios propuestos	Descripción general Modo de operación
<h3>Modo de operación</h3> <div style="display: flex; justify-content: space-around; align-items: center;"> Timer Preload Register (TMRPRELOAD) Cuenta inicial Timer Counter Register (TMRCOUNTER) Cuenta actual </div> <p>1. Se escribe en el registro TMRPRELOAD el valor inicial de la cuenta</p> <p>2. En el siguiente ciclo, se vuelve el valor de TMRPRELOAD en el registro TMRCOUNTER</p> <p>3. Cada ciclo de ejecución se decremente en una unidad el valor de TMRCOUNTER</p> <p>4. Cuando el valor de TMRCOUNTER llega a 0 se dispara la interrupción</p> <p>5. En el siguiente ciclo se vuelve a cargar la cuenta inicial de TMRPRELOAD en TMRCOUNTER</p>	
Sistemas Operativos Interrupciones y entrada/salida	
7 / 15	

Probar el siguiente ejemplo:

```
MOV A, 20
OUT 3
HLT
```

A continuación, ir ejecutando paso a paso las instrucciones para ver cómo se decrementa el valor del contador.

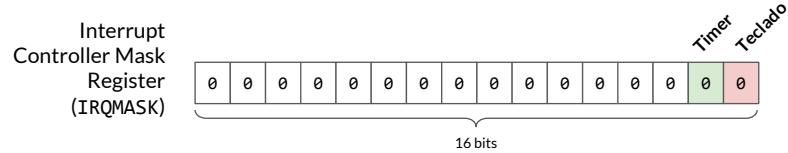
Mapa de entrada/salida El temporizador periódico El controlador de interrupciones Manejadores de interrupción Ejercicios propuestos	Descripción general Registros Habilitación de interrupciones
<h3>El controlador de interrupciones</h3> <p>Descripción general</p> <pre> graph LR subgraph Controller [Controlador de interrupciones] direction TB C[IRQ0] --- T1[Teclado numérico] C[IRQ1] --- T2[Temporizador (timer)] C[IRQ2] --- T3[...] C[IRQ15] --- T15[...] end T1 --- Controller T2 --- Controller T3 --- Controller T15 --- Controller Controller --- IRQ[IRQ] IRQ --- Processor[Procesador] </pre> <ul style="list-style-type: none"> • Permite controlar hasta 16 fuentes de interrupción distintas • Las fuentes se pueden habilitar (desenmascarar) o deshabilitar (enmascarar) de forma independiente 	

La CPU tiene una única fuente de interrupción. El controlador de interrupciones es el encargado de activar y desactivar la señal de interrupción general del procesador en función del estado de las fuentes de interrupción. Cada fuente de interrupción propia del control se puede habilitar o deshabilitar de forma independiente. Si se produce una señal de interrupción por una de las fuentes del control, únicamente se propagará hacia el procesador si la fuente correspondiente está habilitada.

Los términos desenmascarar y enmascarar son sinónimos de habilitar y deshabilitar respectivamente.

Registros del controlador de interrupciones

Interrupt Controller Mask Register (IRQMASK)

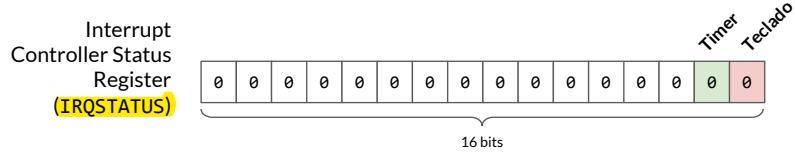


El **registro máscara** del **controlador** de **interrupciones** (IRQMASK) permite **habilitar** y **deshabilitar** las **fuentes** de **interrupción**:

- Si el **bit correspondiente** a la fuente está a **1** \Rightarrow **habilitada**
- Si el **bit correspondiente** a la fuente está a **0** \Rightarrow **deshabilitada**

Registros del controlador de interrupciones

Interrupt Controller Status Register (**IRQSTATUS**)



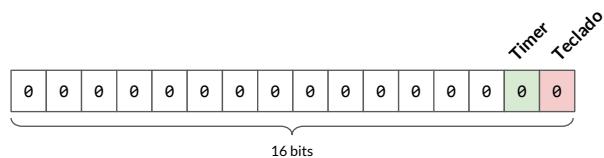
- El **registro** de **estado** del **controlador** de **interrupciones** (**IRQSTATUS**) indica si una **fuente** de **interrupción** está **activada**:
 - Si el bit correspondiente a la fuente está a **1** ⇒ **activada**
 - Si el bit correspondiente a la fuente está a **0** ⇒ **desactivada**
- Es un **registro de sólo lectura**
- **Para que se propague la interrupción, la fuente debe estar activada y habilitada**

Registros del controlador de interrupciones

End of Interrupt Register (**IRQEOI**)

End of Interrupt

Register
(IRQEOI)



- El **registro de fin de interrupción (IRQEOI)** se emplea para indicar al **controlador** que ya ha atendido la **interrupción**
- Es **necesario escribir un 1** en el **bit correspondiente** a la **fuente** para **desactivar la interrupción**
 - Si la **interrupción** queda **correctamente desactivada**, el **bit** correspondiente del **IRQSTATUS** quedará a **0**

Mapa de entrada/salida El temporizador periódico El controlador de interrupciones Manejadores de interrupción Ejercicios propuestos	Descripción general Registros Habilitación de interrupciones
<h3>Habilitación de interrupciones</h3> <ol style="list-style-type: none">1. Poner a 1 los bits del registro IRQMASK correspondientes a la/s fuente/s que se desean habilitar2. Habilitar las interrupciones en el procesador<ul style="list-style-type: none">➤ La CPU implementa dos instrucciones para habilitar y deshabilitar las interrupciones de forma global	
<p>Instrucción STI</p> <p>Habilita las interrupciones en el procesador</p> <p>Instrucción CLI</p> <p>Deshabilita las interrupciones en el procesador</p>	

Por ejemplo, para habilitar la interrupción del *timer* periódico habría que codificar las siguientes instrucciones:

```
MOV A, 2  
OUT 0  
STI
```

El procesador siempre comienza con las interrupciones deshabilitadas.

El estado de las interrupciones se puede comprobar en el bit M (Interrupt Mask Bit) del Registro de Estado.

Mapa de entrada/salida El temporizador periódico El controlador de interrupciones Manejadores de interrupción Ejercicios propuestos	Descripción general Marco de interrupción Procesamiento de una interrupción
<h2>Manejadores de interrupción</h2> <p>Descripción general</p> <ul style="list-style-type: none"> • El manejador es la rutina que se ejecuta cuando se dispara una interrupción • Cuando se produce la interrupción, el procesador realiza las siguientes operaciones: <ul style="list-style-type: none"> ○ Si la CPU estaba en modo usuario \Rightarrow modo supervisor ○ Guarda el marco de interrupción (interrupt frame) ○ Modifica el registro puntero de instrucción  <pre> graph LR A[Instruction Pointer] --> B[Valor previo] B --> C[0x0003] </pre>	
Sistemas Operativos Interrupciones y entrada/salida	
13 / 15	

Marco de interrupción

- El **marco de interrupción** contiene la **información necesaria** para poder **retornar al estado previo** al disparo:
 - El **valor previo del puntero de instrucción**
 - El **valor previo del puntero de pila**
 - El **valor previo del registro de estado**
- El **marco se almacena en la pila**

El valor de los registros de propósito general no se guarda, es necesario hacerlo explícitamente en el manejador



Comentar paso a paso el Ejemplo 4 (Sample 4) del simulador. Ver cómo se carga el marco de interrupción.

Ejercicios propuestos

Implementar un programa que haga lo siguiente:

- Instalar un manejador de interrupción para el temporizador
 - El código principal debe escribir un valor fijo de 8 bits en todas las posiciones del display visual de forma cíclica
 - Cuando salte la interrupción, el manejador tiene que incrementar el valor en 10 unidades y retornar la ejecución al código principal
- El valor fijo ha de ser almacenado en una posición fija de memoria
➤ El valor inicial a escribir es 0 Decimal

MOV 0x30 - 0x39 - 0x3A