Arquitectura de Computadores

2º Curso – Grado en Ingeniería Informática

Tema 1.
Introducción a los computadores.

Contenido

1. Concepto de Computador
2. Arquitectura de Von Neumann
3. Arquitectura y Organización de un computador
4. Niveles de un Computador
5. Historia de los Computadores
6. Computadora mejorada

Bibliografía

1. H. Taub "Circuitos Digitales y Microprocesadores"
2. Pedro de Miguel Anasagasti "Fundamentos de los Computadores"

Introducción

- Computador u Ordenador
  - Máquina que procesa información de forma automática.
- Procesar
  - Manipulaciones o Transformaciones que se aplican sobre la información (datos) para obtener la solución de un problema determinado.
- Tecnología
  - Elementos físicos que constituyen un computador
  - Tipos
    - Tecnología Electrónica
    - Tecnología Mecánica
    - Tecnología Óptica
- Estructuras
  - Uniones lógicas de elementos tecnológicos que realizan una tarea conjunta.

Clasificación de los Computadores

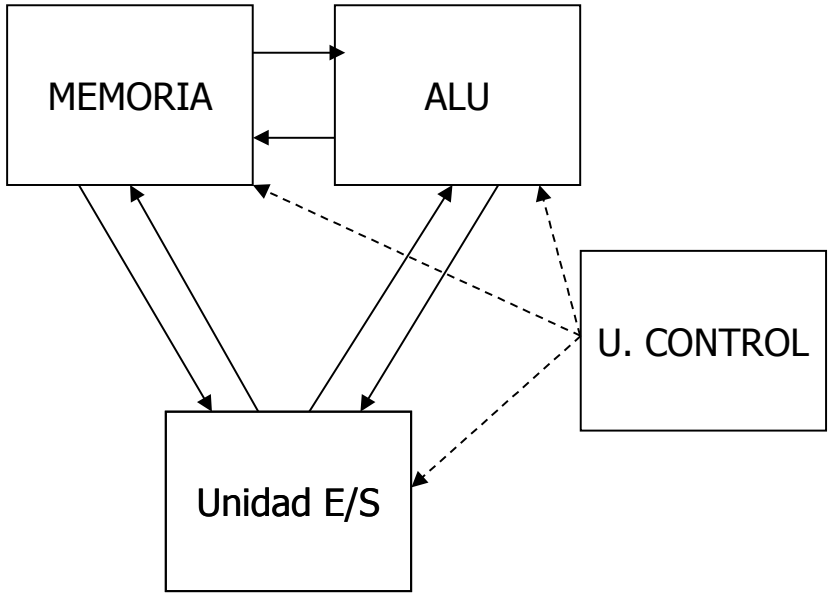
- Computadores Digitales
  - Están basados en tecnología electrónica.
  - Utilizan el sistema binario de representación digital.
  - Los dígitos binarios se representan con valores de tensión eléctrica. Las dos tensiones consideradas se denominan H (*high*) y L (*low*).
  - La asignación de los valores 1 y 0 lógicos es arbitraria, aunque normalmente H suele asociarse a 1 y L a 0.
  - Los bits se pueden asociar en grupos de varios bits para representar números, letras o cadenas de caracteres.
- Computadores Analógicos
  - Trabajan con valores de tensión, cuyo rango está restringido (usualmente -10V .. +10 V).
  - Se debe utilizar factores de escalado y requieren mucha precisión para la manipulación numérica.

Clasificación de los Computadores

- Computadores Digitales vs. Computadores Analógicos
  - Los Computadores Digitales poseen mayor capacidad y facilidad de manejo de la información.
  - Tienen mayor precisión en la representación numérica.
  - Mayor facilidad para aumentar el rango de representación.
  - Mayor facilidad de empleo.
  - Permiten la manipulación de datos e información no numérica.
  - Sin embargo, los computadores analógicos son utilizados para ciertas aplicaciones específicas, debido al paralelismo y su alta velocidad de cálculo.

Arquitectura de Von Neumann

- Arquitectura de Von Neumann
  - En 1945, Von Neumann propuso un modelo de computación que ha constituido, incluso en la actualidad, la base de la arquitectura de los computadores digitales.
  - Está compuesto por 4 unidades:
    - Unidad de Memoria Principal
    - Unidad de Entrada/Salida
    - Unidad Aritmético-Lógica
    - Unidad de Control
  - La CPU suele incluir:
    - Unidad Aritmético-Lógica
    - Unidad de Control



Arquitectura de Von Neumann

- La arquitectura de Von Neumann está diseñada para procesar información: DATOS + PROCESAMIENTO.
- La arquitectura de Von Neumann no especifica el tipo de dato con el que se trabaja.
  - BCD
  - Byte
  - Word
  - Entero con/sin signo
  - Reales de mayor o menor precisión
- El tipo de procesamiento vendrá indicado por instrucciones:
  - Aritméticas
  - Lógicas
  - Transferencia de datos
  - Control de flujo (saltos, bifurcaciones, etc.)
  - Otras
- La secuencia de instrucciones se denomina programa.
- Tanto los datos como el programa se almacenarán en el bloque de Memoria Principal.

Arquitectura de Von Neumann

- Tres conceptos:
  - Instrucciones y datos almacenados en una misma memoria unificada.
  - El contenido de memoria se direcciona por *localidad*: La memoria se accede por la posición que ocupa y no por el tipo de dato que almacena.
  - La ejecución de las instrucciones es secuencial (tras la ejecución de una instrucción se ejecuta la que se encuentra en la siguiente posición de programa). Existen instrucciones que rompen esta secuencialidad.

Arquitectura de Von Neumann: Memoria Principal

- Compuesta por un conjunto de celdas de almacenamiento agrupadas en bloques del mismo tamaño (*palabra*).
- El acceso a cada palabra tiene asociada un número único, llamada *dirección*.
- La Memoria Principal viene caracterizada por 2 parámetros: Tamaño de memoria (número de direcciones) y anchura de palabra (número de bits que componen cada palabra).
- Las operaciones posibles sobre la memoria son:
  - Lectura
  - Escritura
- En la Arquitectura de Von Neumann:
  - Los datos y las instrucciones se almacenan en la misma unidad de memoria principal.

Arquitectura de Von Neumann: Unidad Aritmético-Lógica

- La ALU posee una serie de microoperaciones que son capaces de tomar datos y aplicarles operaciones elementales.
- Para que la ALU sea efectiva, incorpora registros que almacenan los operandos y guardan el resultado de la operación.
- *Banco de Registros*:
  - Registros de propósito general
  - Registros específicos
  - Registros de control
  - Registros de estado
- El tamaño de los registros es uno de los factores que limita los tipos de datos con los que opera la computadora.
- Los datos se leen de memoria, se almacenan temporalmente en los registros.

Arquitectura de Von Neumann: Unidad de Control

- La Unidad de Control es la encargada de manejar todas las demás unidades de la computadora, de manera que trabajen coordinadamente para realizar tareas más complejas.
- Ésta toma instrucciones de la memoria principal, las decodifica y prepara las señales para las diferentes unidades funcionales y bloques que tendrá que generar en los diferentes ciclos en los que se descompone la misma.
- Generará las señales necesarias para acceder y manejar:
  - Memoria Principal (lectura o escritura)
  - ALU (diversas operaciones), Banco de Registros (lectura o escritura)
  - E/S (diversas operaciones)
- Para poder llevar la cuenta de la instrucción del programa que se tendrá que ejecutar, necesita un registro apuntador, denominado genéricamente *Contador de Programa*.

Arquitectura de Von Neumann: Unidad de Entrada/Salida

- Un computador existe dentro de un mundo real y necesita tomar datos desde el exterior y debe mostrar resultados a los usuarios que lo utilizan.
- La Unidad de E/S realiza las transferencias de información con los *periféricos*.
- La Unidad de E/S es capaz de realizar cargas de información desde los discos externos (también llamados *Memoria Secundaria*) directamente a Memoria Principal. Esto ocurre en la carga de programas y datos desde disco.
- También es la encargada de la visualización por pantalla, impresión por papel mediante la impresora, acceso a la red, etc.

Arquitectura de Von Neumann: Buses de Interconexión

- Las diversas unidades se interconectan entre sí mediante buses. Estos buses permiten transferir instrucciones y datos entre las distintas unidades.
- En particular hay 3 buses:
  - Bus de Direcciones
  - Bus de Datos
  - Bus de Control
- El interconexionado de los buses es específico.
  - Interconexionado general (desde cualquier unidad o registro se puede llegar directamente a cualquier otra unidad o registro).
  - Interconexionado limitado (se permiten solo conexiones de algunas unidades funcionales o registros con otras unidades o registros específicos y no con todos)

Arquitectura de Von Neumann: CPU

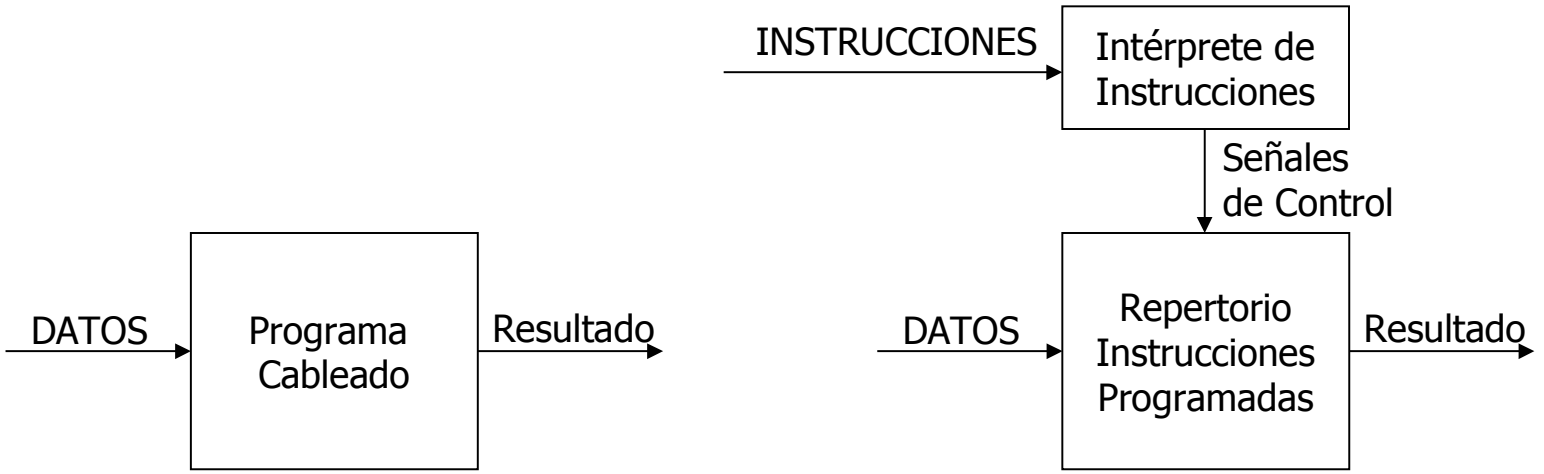
- La Unidad Central de Procesamiento (CPU) es el conjunto formado por la ALU, con sus registros internos y la Unidad de Control: Es el bloque encargado de la ejecución de las instrucciones.
- También suele incluir la Unidad de E/S.
- Después de la aparición de los circuitos integrados, la CPU estaba formada por varios bloques, que terminaron siendo integrados en un único circuito denominado genéricamente *Microprocesador*.
- La capacidad de la CPU viene medida por mucho factores:
  - MIPS
  - MFLOPS
  - Frecuencia del procesador

Programa cableado vs. Funciones preprogramadas

- Un determinado problema se puede resolver mediante una configuración de componentes lógicos básicos (puertas lógicas combinacionales).
- Si el algoritmo o programa se implementa mediante puertas lógicas, se dice que tiene el *programa cableado*.
  - No es modificable => Poca flexibilidad
  - Es rápido
  - Solución *ad-hoc* o muy específica
- Otra solución, construir un conjunto de funciones aritméticas y lógicas de propósito general mediante las que conformar el programa.
  - Programas modificables => Alta flexibilidad
  - Lento
  - Solución de propósito general

Programa cableado vs. Funciones preprogramadas

- Dos enfoques:
  - Programación en **Hardware**:
    - Programación Cableada, dependen de los componentes y de cómo estén interconectados.
  - Programación en **Software**:
    - El Hardware tiene una determinada interconexión que permite ser utilizado de manera genérica.



Programa cableado vs. Funciones preprogramadas

- El sistema es capaz de hacer una cierta cantidad de funciones (posee el hardware necesario para realizar varias operaciones aritmético-lógicas).
- Mediante señales de control, el sistema puede seleccionar qué operación realizar.
- *¿Cómo proporcionar dichas señales de control?*
  - Secuencia de pasos.
  - Cada paso efectúa alguna operación sobre los datos: señales que hay que activar y desactivar
  - Asociar un código único a cada conjunto de señales de control
  - Construir un bloque que active dichas señales

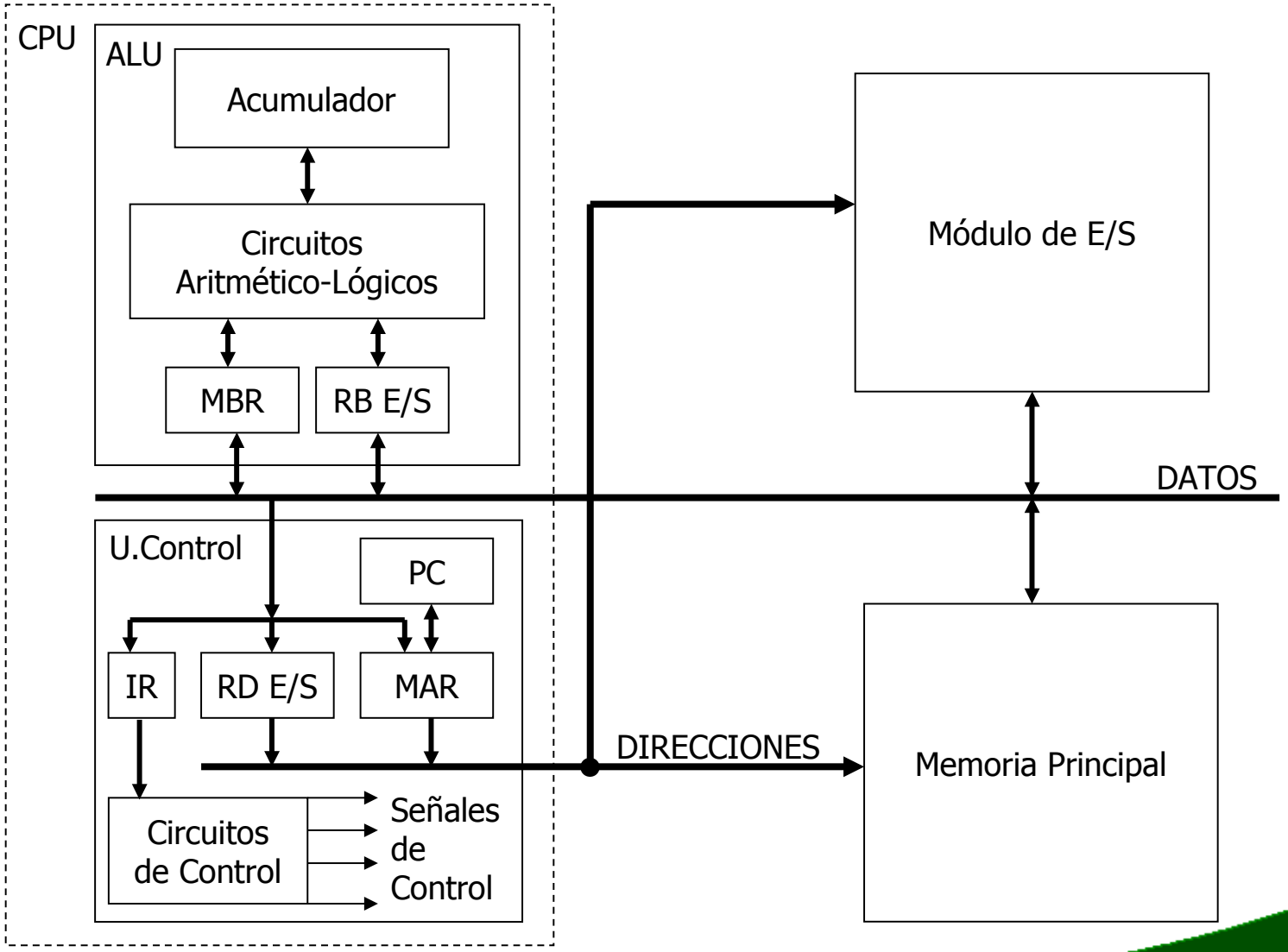
Programa cableado vs. Funciones preprogramadas

- Con el nuevo hardware obtenido, para cada programa sólo es necesario proporcionar una secuencia de códigos, en vez de cambiar la configuración de su conexionado.
  - Cada código es una instrucción.
  - El hardware incluirá un bloque que interpretará cada instrucción y generará las señales de control.
  - A este nuevo método de programación se le denomina *software*, que corresponde a una secuencia de instrucciones.
- Estos módulos corresponderán a la ALU y a la Unidad de Control.

Programas: Instrucciones y Datos

- En la Memoria Principal se almacenan tanto instrucciones como datos.
- Es tarea del programador indicar cómo se debe tratar cada posición de memoria:
  - Posición de memoria, ¿contiene una instrucción o un dato?
- Si es un dato, no hay que interpretarlo. Solo manejarlo conforme indiquen las instrucciones que hagan uso de él.
- Si es una instrucción, hay que interpretarla y generar el conjunto de señales de control.

Diagrama de la Computadora Básica según la Arquitectura de Von Neumann

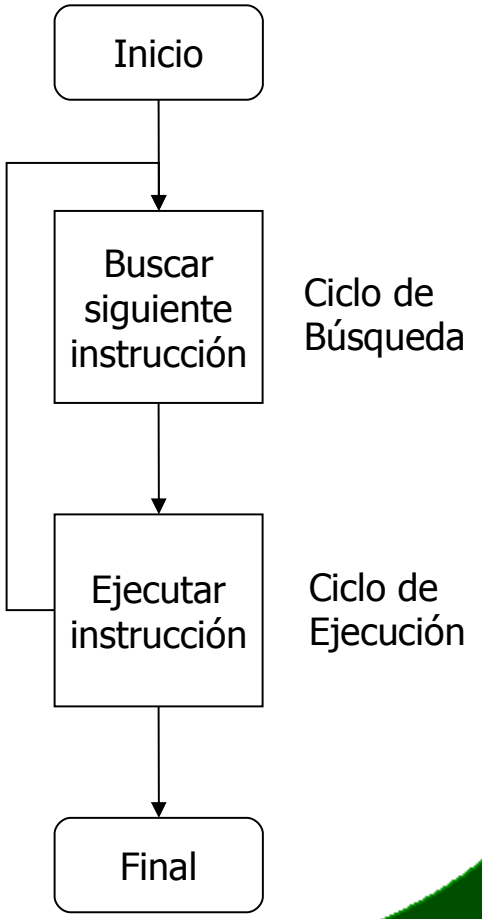


Registros de Control en la Computadora Básica de Von Neumann

- **MBR** (*Memory Buffer Register*)
  - Contiene el dato que se va a escribir en la memoria, o que se lee de ésta.
- **MAR** (*Memory Address Register*)
  - Especifica la dirección de memoria de la palabra que se va a escribir o leer.
- **RBE/S** (*Registro Buffer para Entrada/Salida*)
  - Semejante al MBR. Se utiliza para intercambiar datos entre E/S y la CPU.
- **RDE/S** (*Registro Dirección para Entrada/Salida*)
  - Es similar al MAR. Especifica un dispositivo de E/S.
- **IR** (*Instruction Register*)
  - Contiene el código de operación de la instrucción que se está ejecutando.
- **PC** (*Program Counter*)
  - Contiene la dirección de memoria que contiene la siguiente instrucción.
- **Ac** (*Accumulator*)
  - Almacena temporalmente los operandos y los resultados de las operaciones de la ALU.

Funcionamiento de la Computadora

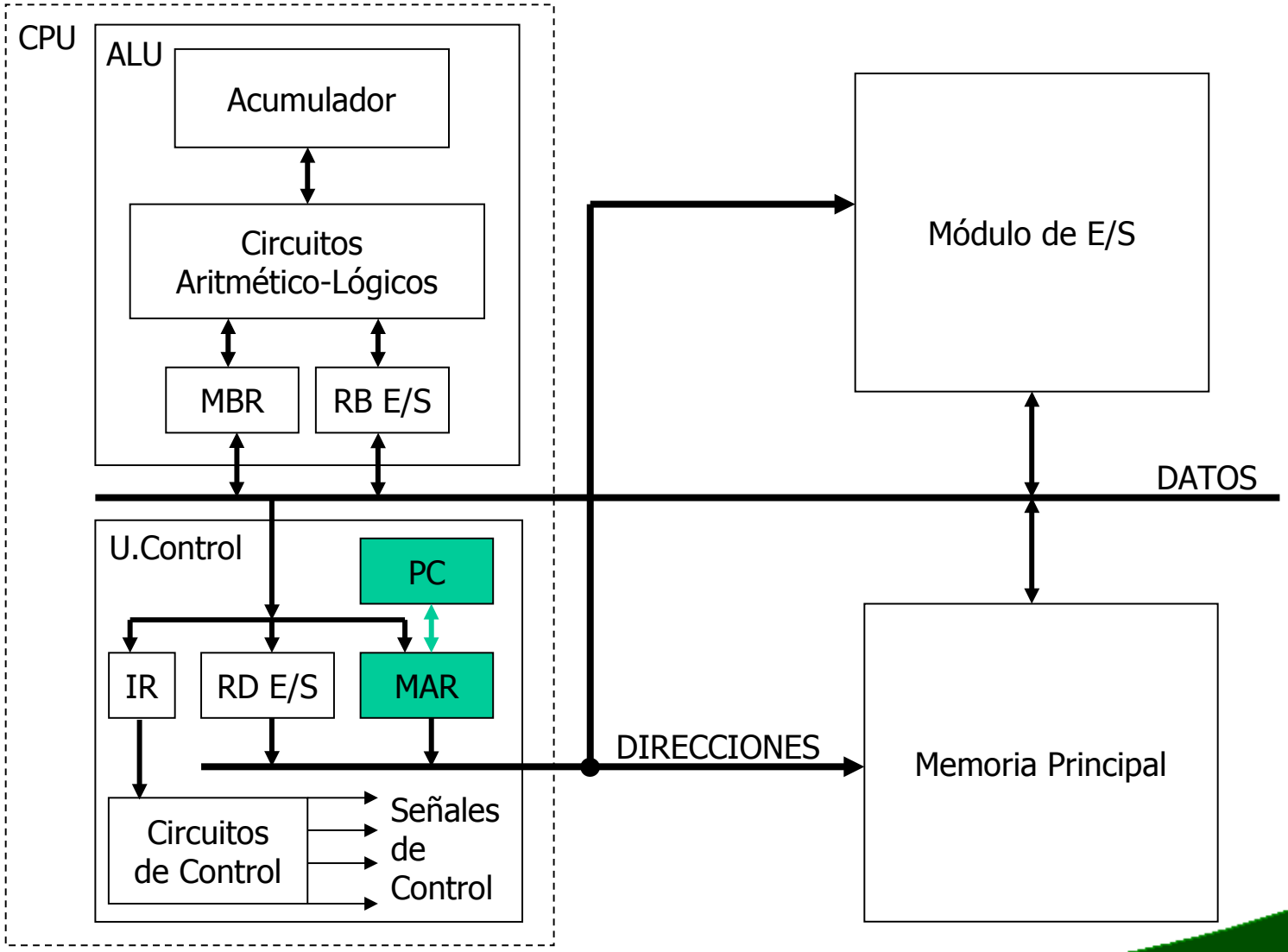
- La función básica de un computador es ejecutar un programa.
- Un programa es un conjunto de instrucciones almacenadas en memoria.
- El proceso para ejecutar una instrucción se denomina ***ciclo de instrucción***. Se compone de dos fases, denominadas ***ciclo de búsqueda*** y ***ciclo de ejecución***.
  - El ciclo de instrucción empieza con la búsqueda (lectura) de la instrucción de la memoria y termina con la ejecución de la misma.
  - La ejecución de un programa consiste en la repetición del proceso anterior, es decir de múltiples ciclos de instrucción.



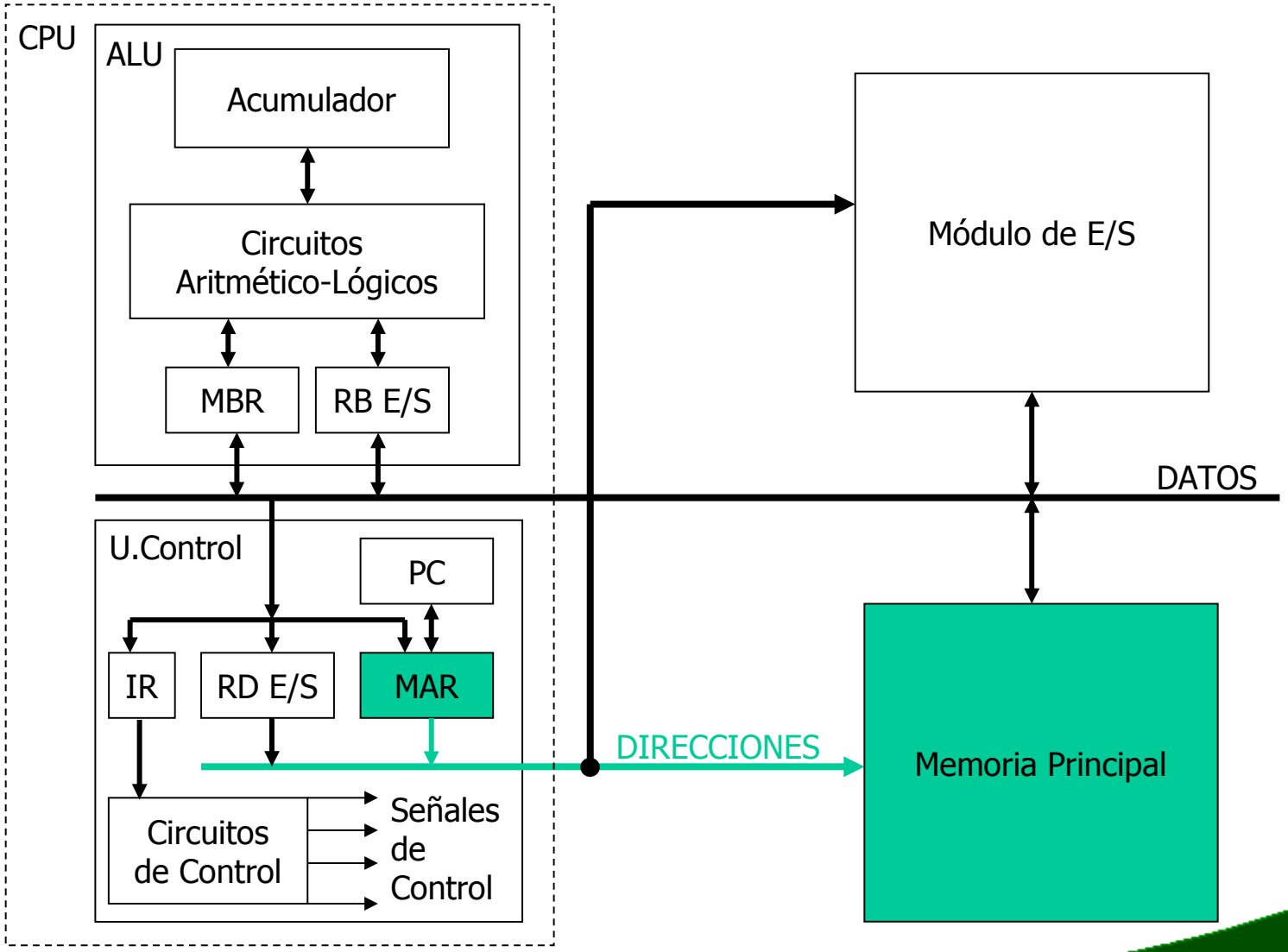
Ciclo de Búsqueda

- El inicio de cada instrucción es igual siempre:
  - Buscar en la memoria la siguiente instrucción.
  - Traer la instrucción desde memoria a la Unidad de Control.
  - Decodificar la instrucción.
- El algoritmo que realiza el ciclo de búsqueda sería:
  - Transferir el contenido de PC a MAR.
  - Incrementar PC.
  - Enviar la dirección de MAR a la Memoria Principal. Leer la posición de memoria Memoria[MAR].
  - Poner el resultado en el bus de datos y transfiere de memoria a MBR.
  - Transferir de MBR a IR.

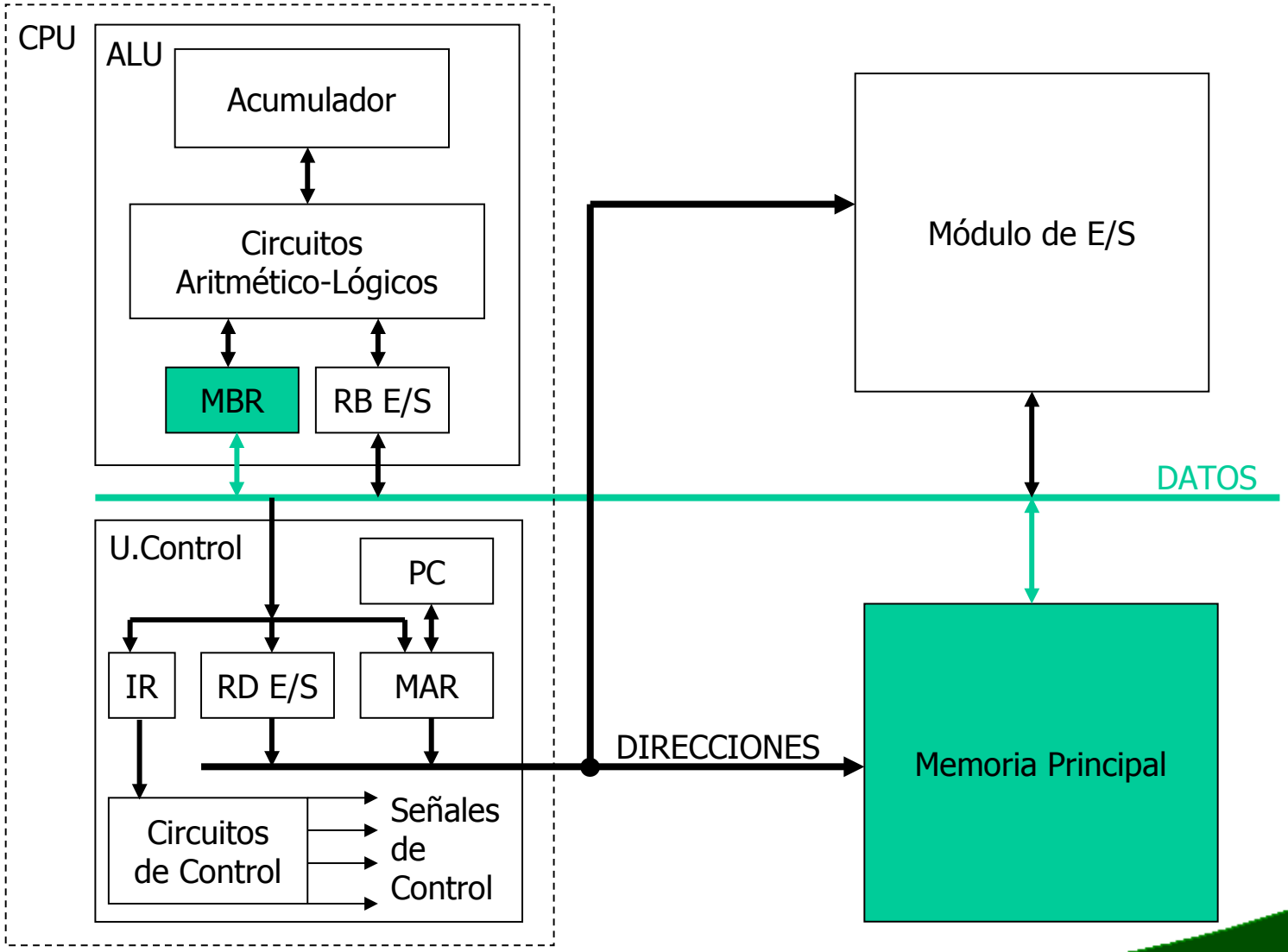
Ciclo de Búsqueda



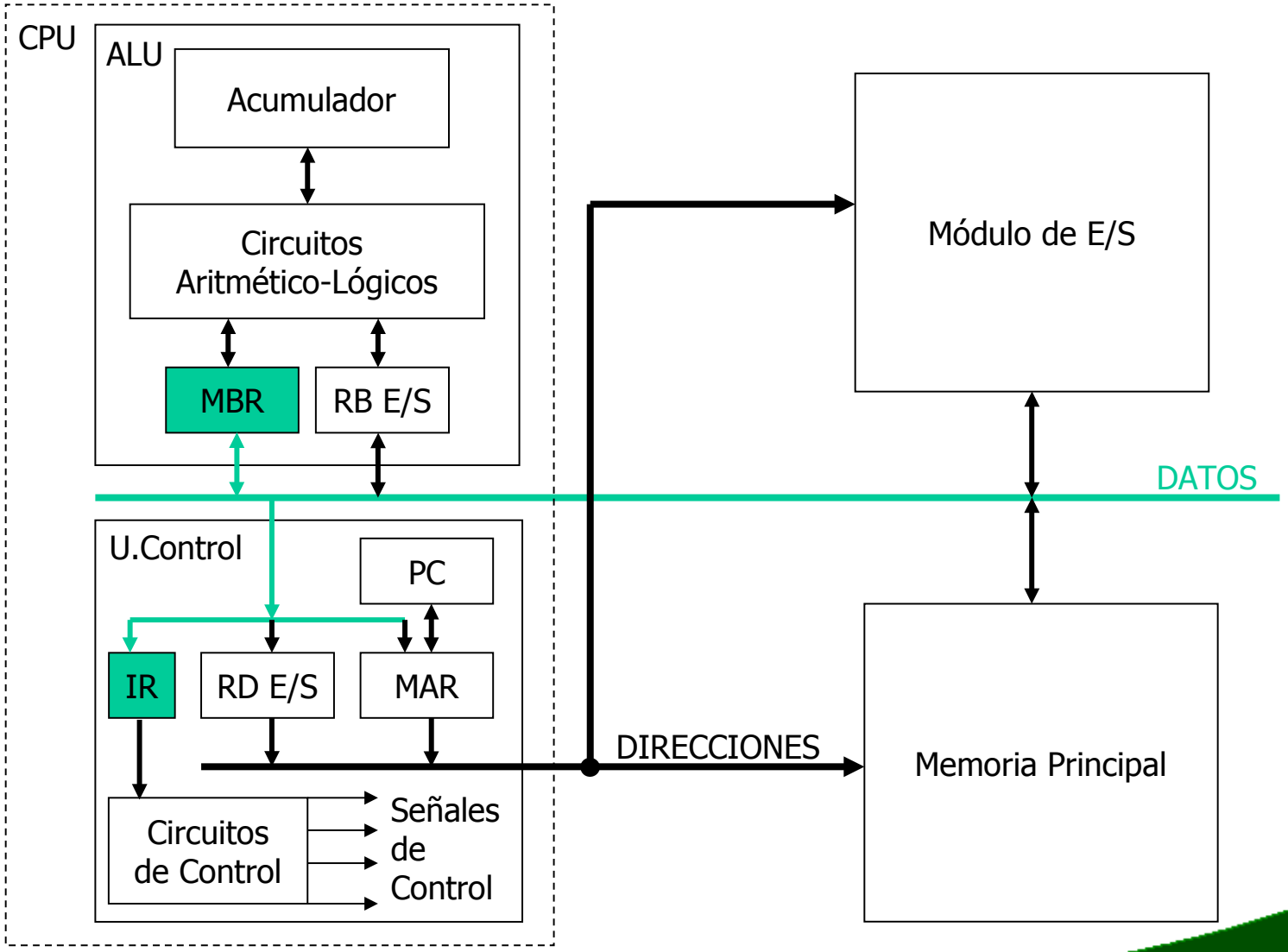
Ciclo de Búsqueda



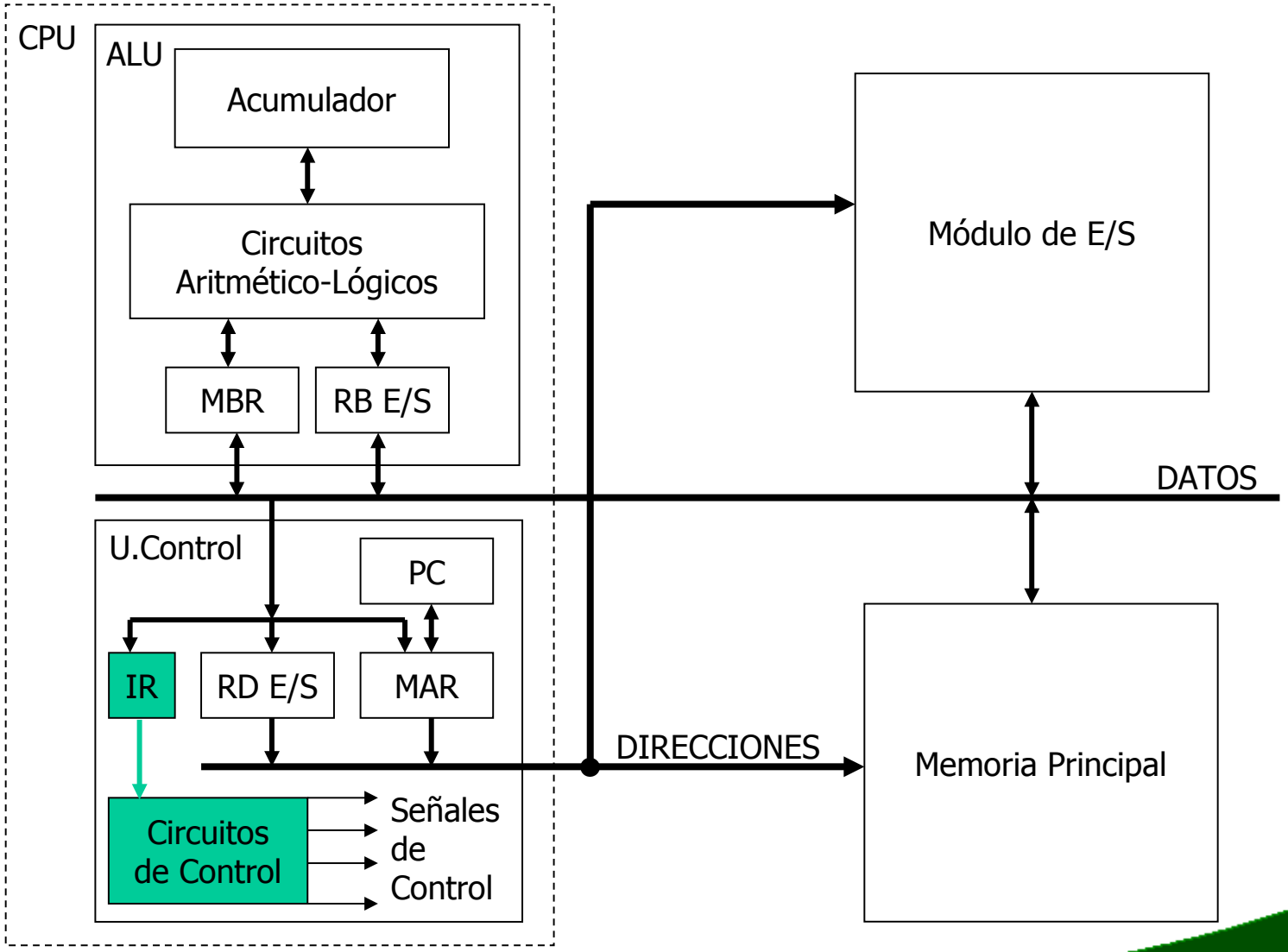
Ciclo de Búsqueda



Ciclo de Búsqueda



Ciclo de Ejecución

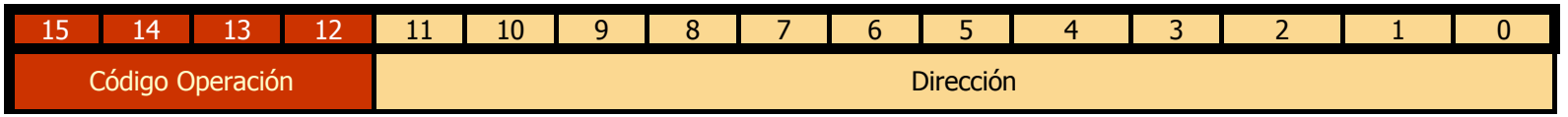


Ciclo de Ejecución

- Cada instrucción se representa con un código binario único.
- El código binario asociado suele estar compuesto por varios campos, que representan el código de operación (que especifica el tipo de operación) y el código de los operandos (que especifican los tipos de operandos de la operación).
- La Unidad de Control interpreta el código de operación, que se puede clasificar en 4 grandes tipos:
  - **Transferencia CPU-Memoria**
  - **Transferencia CPU-E/S**
  - **Procesamiento de Datos**
  - **Control**
    - Una instrucción puede alterar el orden de ejecución de un programa, realizar saltos incondicionales, saltos condicionales, etc.
- Se pueden combinar varios tipos en una única instrucción.

Formato de Instrucción

- El formato de instrucción se divide en dos campos:
  - Código de Operación
  - Código de Operando
- Ejemplo:
  - Supongamos que la palabra tiene un tamaño 16 bits.
  - Utilizaremos 4 bits para el código de operación, por lo que es posible tener 16 códigos de instrucción diferentes.
  - Quedarán libres 12 bits, por lo que se podrán direccionar 2^{12} = 4096 palabras (4K).



Ejemplo de Ejecución

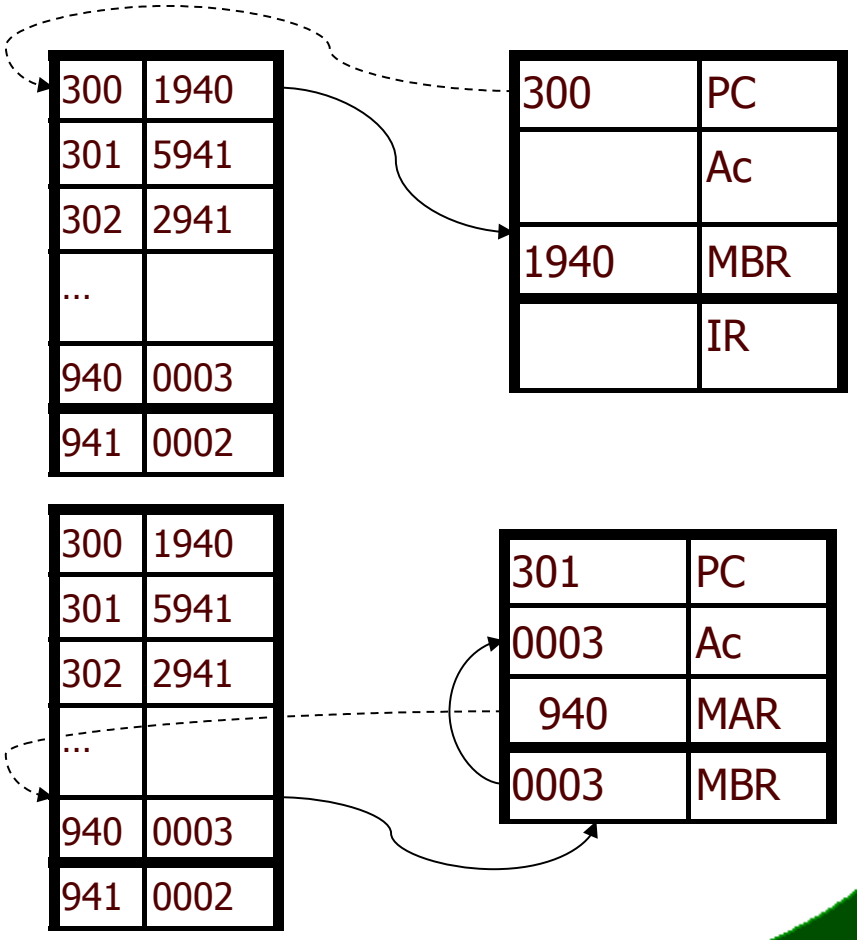
- Programa:
  - Suma el contenido de la palabra de memoria 940_{16} con el contenido de la palabra de memoria 941_{16}
  - Almacena el resultado en la posición 941_{16}
- Sea el repertorio códigos instrucción:
  - 0001 => Carga la posición de memoria en Ac.
  - 0010 => Almacenar el Ac en la posición de memoria.
  - 0101 => Sumar el contenido de la posición de memoria con el Ac.

- El contenido de la memoria sería:

300	1940	0001|940
301	5941	0101|941
302	2941	0010|941
…		
940	0003	0003
941	0002	0002

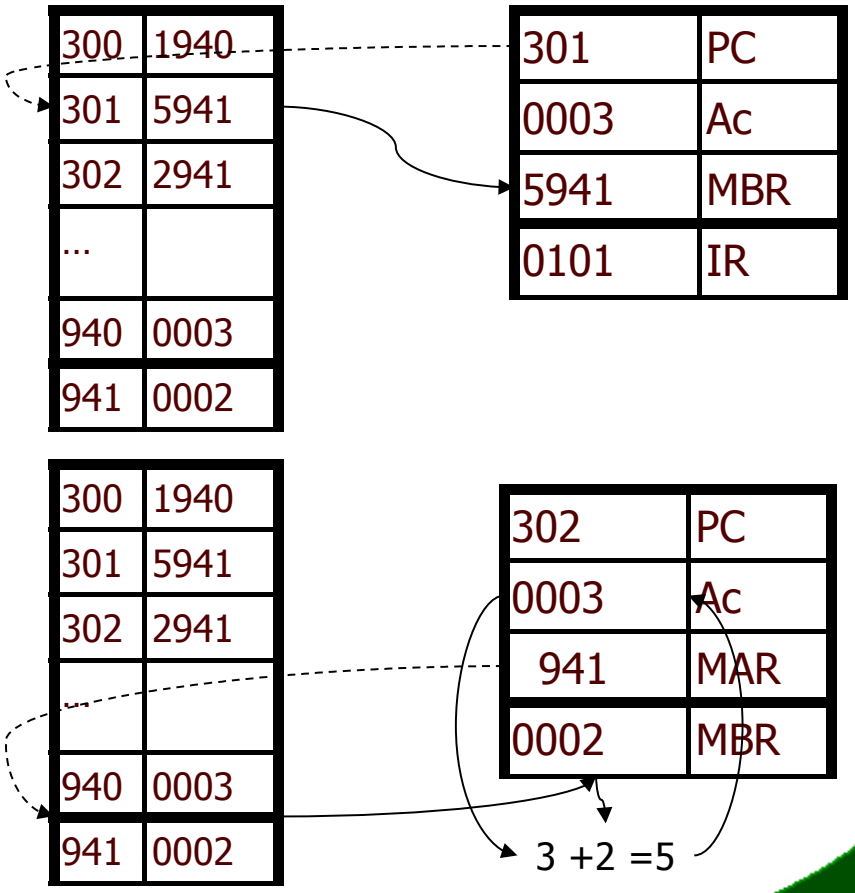
Ejemplo de Ejecución

- **Ciclo de Búsqueda 1º**
  - El programa comienza en la posición 300_{16} (el contador de programa, PC, estará inicializado a 300h).
  - Se transfiere el contenido de PC al MAR.
  - Se accede a la posición de memoria indicada por MAR.
  - Se almacena el dato de la memoria en el MBR.
  - Se transfiere del MBR al registro IR.
- **Ciclo de Ejecución 1º**
  - La operación que se desea realizar tiene código 0001b (Carga una posición de memoria en Ac). La dirección viene indicada en los 12 bits inferiores de la instrucción.
  - Carga MAR con los 12 bits inferiores de MBR (940h).
  - Lee la posición de memoria indicada por MAR y el valor lo almacena en MBR.
  - Carga el valor de MBR en Ac.
  - Se incrementa PC en 1.



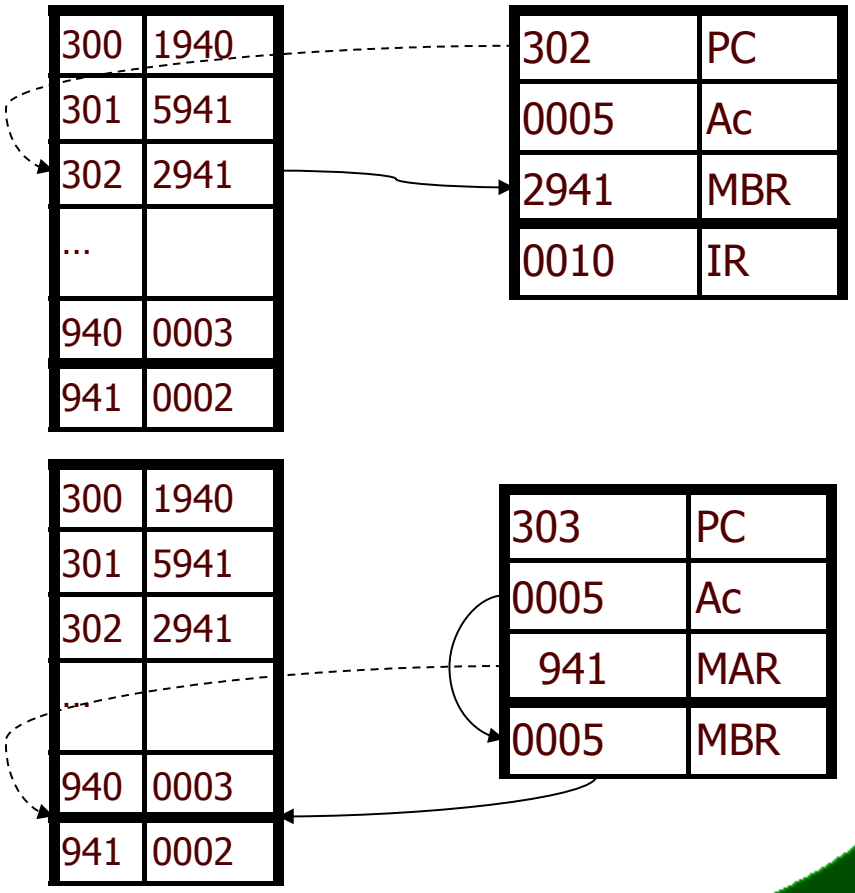
Ejemplo de Ejecución

- **Ciclo de Búsqueda 2º**
  - El programa continúa en la posición 301h (PC contendrá 301h).
  - Se transfiere el contenido de PC al MAR.
  - Se accede a la posición de memoria indicada por MAR.
  - Se almacena el dato de la memoria en el MBR.
  - Se transfiere del MBR al registro IR.
- **Ciclo de Ejecución 2º**
  - La operación que se desea realizar tiene código 0101b (Sumar el contenido de Ac con una dirección de memoria).
  - Carga MAR con los 12 bits inferiores de MBR (941h).
  - Lee la posición de memoria indicada por MAR y el valor lo almacena en MBR.
  - Suma el valor de MBR y el valor de Ac y lo almacena en Ac.
  - Se incrementa PC en 1.



Ejemplo de Ejecución

- **Ciclo de Búsqueda 3º**
  - El programa continúa en la posición 302h (PC contendrá 302h).
  - Se transfiere el contenido de PC al MAR.
  - Se accede a la posición de memoria indicada por MAR.
  - Se almacena el dato de la memoria en el MBR.
  - Se transfiere del MBR al registro IR.
- **Ciclo de Ejecución 3º**
  - La operación que se desea realizar tiene código 0010b (Almacenar el contenido de Ac en una dirección de memoria).
  - Carga MAR con los 12 bits inferiores de MBR (941h).
  - Transfiere el valor de Ac a MBR.
  - Almacena en a posición de memoria indicada por MAR el valor que hay en MBR.
  - Se incrementa PC en 1.



Lenguajes de programación

- Programación a bajo nivel:
  - Programación en ensamblador
    - Cada instrucción es un mnemónico (nombre simbólico).
    - Próximo al lenguaje máquina.
    - El programa traductor/compilador se denomina *ensamblador*.
    - Cada computador tiene un lenguaje ensamblador diferente.
- Programación a alto nivel:
  - Programación en lenguajes de alto nivel
    - Instrucciones complejas.
    - Tipos de datos compuestos.
    - Programación más cercana al pensamiento humano y más alejado de la máquina.
    - La sintaxis y los elementos (tipos de datos, operandos, operadores, sentencias) son independientes del computador en el que se ejecuta => Portabilidad a nivel de código fuente.

Arquitectura y Organización

- Arquitectura de un Computador
  - Son las características del sistema que ve un programador que trabaje en lenguaje ensamblador.
  - Características:
    - Juego de instrucciones del computador.
    - Tipos y formatos de operandos.
    - Mapa de memoria y de E/S.
    - Modelo de Ejecución.
  - Clasificación según los mapas de memoria y de E/S
    - Arquitectura de Von Neumann
    - Arquitectura Harvard
- Organización de un Computador
  - También denominado **estructura interna**. Se refiere a las unidades que tiene el computador y a la forma que se conectan entre sí para determinar una arquitectura específica.
  - Características transparentes al programador:
    - Señales de Control
    - Interfaces Computadora/Periféricos
    - Tecnología

Niveles de Estudio del Computador

- Un Computador se puede estudiar desde varios puntos de vista:
- Descripciones más habituales:
  - Niveles estructurales de Bell y Newell
  - Nivel de interpretación de Levy
  - Niveles conceptuales de Blaauw

Niveles estructurales de Bell y Newell

- Se basa en la naturaleza jerárquica de un computador.
- Se divide el computador en niveles estructurales, cada uno de los cuales se compone de bloques o componentes que se construyen con bloques o componentes de un nivel inferior.
- Un diseñador o analista solo necesita estudiar un nivel particular en un momento dado.
  - En cada nivel, solo interesa su estructura y su función.
- Bell y Nevell dividen el estudio del computador en 5 niveles:
  - Nivel de componentes.
  - Nivel de circuito electrónico.
  - Nivel de circuito digital.
  - Nivel de transferencia entre registros
    - (RTL *Register Transfer Logic*).
  - Nivel de Intercambio Memoria-Procesador
    - (PMS *Processor Memory Switch*)

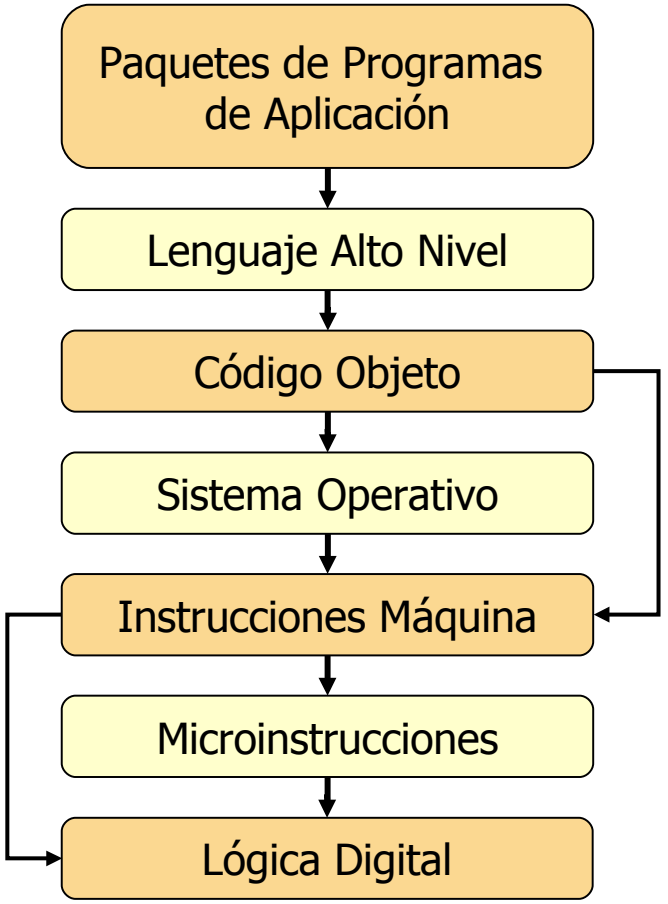
Niveles estructurales de Bell y Newell

Nivel	Primitivas	Reglas de Combinación	Elementos Complejos
Componentes	Semiconductor, metal, óxido	Electrónica, Física del estado Sólido	Diodo, transistor, resistencia, condensador
Circuito Electrónico	Diodo, transistor, resistencia, condensador	Leyes de Kirchoff, características de transferencia	Puertas Lógicas, Biestables
Circuito Digital	Puertas Lógicas, Biestables	Álgebra de Boole, Minimización, Teoría de autómatas	Multiplexores, Decodificadores, Registros de desplazamiento, contadores
RTL	Multiplexores, Decodificadores, Registros de desplazamiento, contadores	Lenguajes de transferencia entre registros, microprogramación	Memoria, ALU, Unidad de Control
PMS	Memoria, ALU, Unidad de Control	Estructura, Organización y Arquitectura de Computadores	Ordenador

Niveles de Interpretación de Levy

- La clasificación de Levy se hace desde un punto de vista funcional. Cada nivel se considera como un intérprete que recibe unas instrucciones de un cierto tipo y actúa de acuerdo a ellas:
  - Nivel de lógica digital
  - Nivel de microinstrucciones.
    - Firmware.
  - Nivel de instrucciones de máquina.
  - Nivel de sistema operativo
    - Conjunto de programas para la explotación del computador.
  - Nivel de código objeto.
    - Traducción (lenguaje ensamblador).
  - Nivel de lenguaje de alto nivel.
    - Compilación y enlazado.
  - Nivel de paquetes de programas de aplicación.
- El nivel de código objeto puede traspasar el nivel de sistema operativo e interaccionar directamente con el segundo nivel.

Niveles de Interpretación de Levy



Niveles Conceptuales de Blaauw

- Se definen 3 niveles conceptuales:
  - **Arquitectura**.
    - Comportamiento funcional del computador, tal y como aparece para el programador en ensamblador.
    - Conjunto de instrucciones máquina y representaciones usadas por el compilador para manejar la información.
    - Es el ¿QUÉ HACE?
  - **Configuración**.
    - Organización interna del computador a nivel de transferencia entre registros y de flujo de información.
    - Muchas configuraciones diferentes pueden responder a una única arquitectura, pero hay que buscar la más productiva y rápida
    - Es el ¿CÓMO LO HACE?
  - **Realización**.
    - Es la forma en que la configuración se plasma con elementos físicos concretos.
    - Una misma configuración admite múltiples realizaciones.
    - Es el ¿QUIÉN LO HACE?

Historia de los Computadores

- Clasificación según la tecnología utilizada:
  - 1ª Generación (1946-1957): Tubos de Vacío. 40KIPS.
  - 2ª Generación (1958-1964): Transistor. 200KIPS.
  - 3ª Generación (1965-1971): SSI/MSI. 1 MIPS.
  - 4ª Generación (1972-1977): LSI. 10 MIPS.
  - 5ª Generación (1978- …) : VLSI. 100+ MIPS.
- Clasificación según Potencialidad y utilización:
  - PDA / Smartphone / Blackberry.
  - Subnotebook.
  - Microcomputador u Ordenador Personal.
  - Minicomputador.
  - Mainframe.
  - Supercomputador.

Historia de los Computadores

- Según el repertorio:
  - CISC (*Complex Instruction Set Computer*)
    - El conjunto de Instrucciones tiene instrucciones muy complicadas desde el punto de vista computacional.
    - Permiten operandos tanto en registros como en memoria.
    - El tiempo de ejecución puede ir desde 1 ciclo hasta varios cientos.
    - El tamaño de las instrucciones puede varíar.
  - RISC (*Reduced Instruction Set Computer*)
    - El conjunto de Instrucciones tiene instrucciones sencillas.
    - El tiempo de ejecución de la mayoría de instrucciones es uniforme.
    - Todas las instrucciones tienen la misma longitud en bits.
    - Todos los operandos se encuentran en registros.
    - Solo existe 1 instrucción para leer de memoria y 1 instrucción para escribir en memoria.
  - VLIW (*Very Large Instruction Words*)
    - Se agrupan varias instrucciones independientes entre sí en una "superinstrucción" para que se ejecuten concurrentemente.

Historia de los Computadores

- 1ª Generación:
  - Utilizan Tubos de Vacío: Alta disipación, Mucho espacio, Mucho consumo energético.
  - 2 ramas:
    - Gran Bretaña: Alan Turing
    - EEUU: Atanasoff y Berry.
  - ENIAC: Primer computador digital electrónico de propósito general.
    - Resolvía Problemas balísticos.
    - 30 Toneladas de peso, 1400 m^2, 18000 tubos de vacío, 140 KW de consumo eléctrico, 5000 sumas por segundo.
    - Máquina decimal con programación manual.
    - Se utilizó en el diseño de la bomba H.
  - IBM: Primera gran empresa de la industria comercial de computadores.
    - Serie 700/7000: 701, uso científico. 702, aplicaciones comerciales.
    - Programas almacenados en tarjetas perforadas.

Historia de los Computadores

- 2ª Generación:
  - Utilizan transistores (1947, Bell Labs): pequeño, barato, disipa menos potencia, funciona a mayor frecuencia.
  - Memoria: Núcleos de ferrita.
  - Almacenamiento masivo: Disco rígido magnético (Winchester).
  - Mejoras en la Arquitectura: Introducción de ALU y U. Control más complejas, Uso de lenguajes de programación de alto nivel.
  - Computadoras:
    - PDP-1 (DEC).
    - Serie 7000 (IBM). Ejemplo, IBM7094:
      - 32K palabras de 36 bits.
      - Tiempo de ciclo 1.4 μs.
      - 185 códigos de operación.
      - 7 registros índice.
      - Representación de punto flotante en doble precisión.
      - Solape de búsqueda de instrucciones.

Historia de los Computadores

- 3ª Generación:
  - Utilizan circuitos integrados (1958), sobre un substrato de silicio de unos pocos mm2 contenía cientos o miles de transistores:
    - Disminución del tamaño, costo y disipación de potencia.
    - Aumento de la frecuencia de funcionamiento.
    - Aumento de la fiabilidad.
  - Sistemas Operativos (IBM OS/360, IBM MVS, DEC VMS, Bell Lab UNIX)

- Familia de Microcomputadoras: Serie IBM 360.
  - Primera familia de computadores planificada.
    - Conjunto de instrucciones semejante o idéntico.
    - Sistema operativo semejante o idéntico.
    - Aumento de velocidad.
    - Aumento en el número de puertos de E/S.
    - Aumento en el tamaño de memoria.
    - Aumento del costo.
- DEC PDP-8: Primer Minicomputador
  - Tamaño reducido, bajo coste.
  - Todas las computadoras anteriores utilizaban un sistema central de conmutación.
  - PDP-8 utilizaba un sistema en bus (Omnibus, 96 señales de control, direcciones y datos).
  - Todos los componentes son controlables desde la CPU.
  - Permite conectar cualquier módulo al bus y crear diferentes configuraciones.

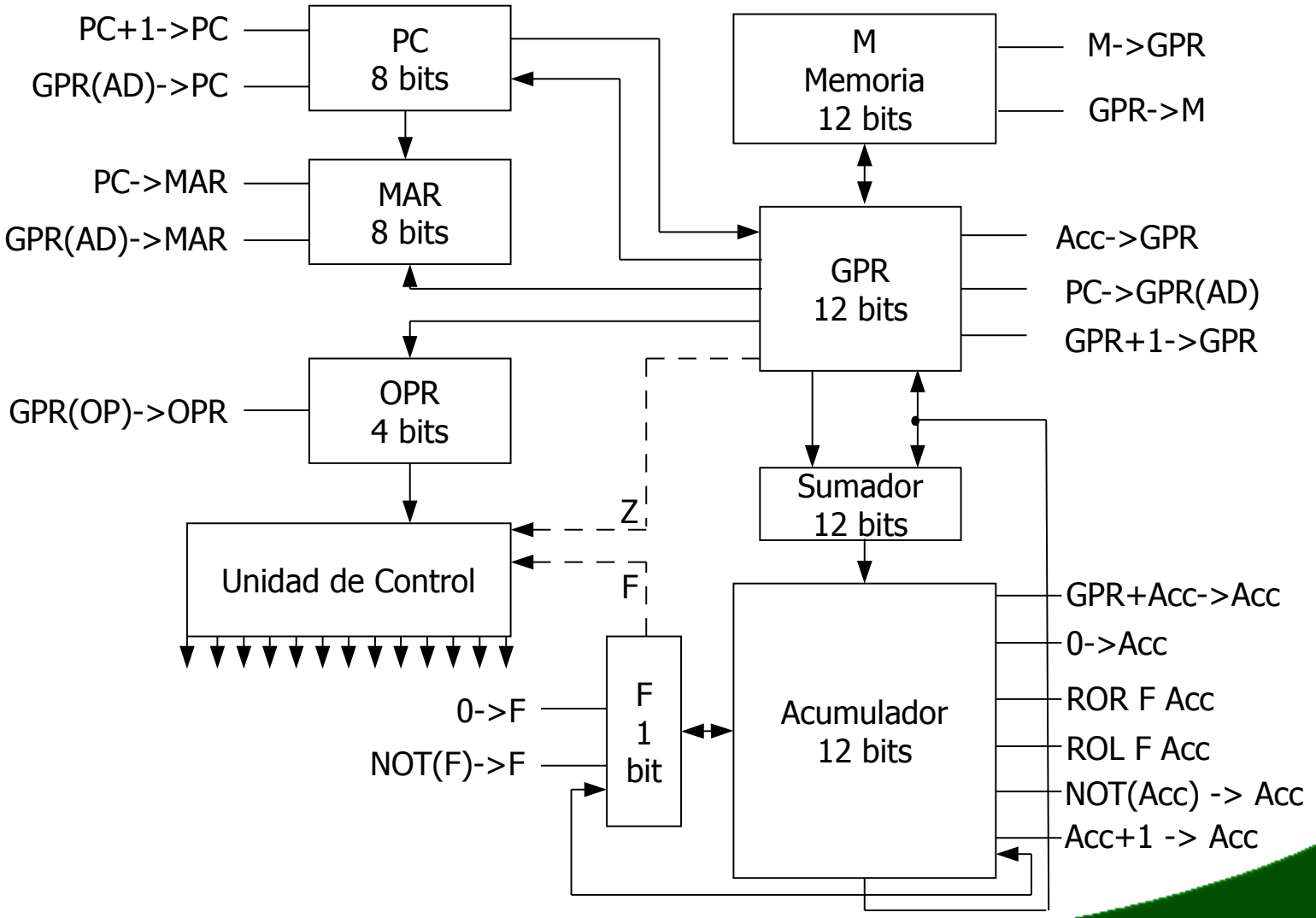
Historia de los Computadores

- 4ª Generación:
  - Se caracteriza por la aparición de las memorias semiconductoras y del microprocesador, es decir la CPU (ALU y U. Control) en un único circuito integrado.
  - Se construyen ordenadores más baratos, de menor tamaño y consumo.
  - Intel fabricó el primer microprocesador, el 4004:
    - Palabra de 4 bits.
    - Sumaba números de 4 bits.
    - Multiplicaba mediante sumas repetitivas.
  - Otro aspecto que introdujo la 4ª generación fue la aparición de los ordenadores personales:
    - 1977: Steve Jobs y Steve Wozniak introducen el Apple II. Ordenador personal de bajo coste y alta fiabilidad.
    - 1981: IBM introduce el IBM-PC basado en el procesador 8088/8086 de Intel. Este ordenador se convirtió en el más vendido de la historia.
  - La 4ª generación generalizó el uso de las redes de computadores.

Historia de los Computadores

- 5ª Generación:
  - La 5ª generación se caracteriza por el uso de circuitos integrados con tecnología VLSI o mayor escala de integración, además de basarse en microprocesadores.
  - Se comienza a utilizar más habitualmente la arquitectura Hardvard (RISC) frente a la Von Neumann (CISC).
  - Se desarrollan *arquitecturas paralelas*, *sistemas multiprocesadores*, *sistemas distribuidos* y *clústeres de computadores*.
  - Se desarrolla la *inteligencia artificial*:
    - *Sistemas expertos* (toman decisiones sobre temas concretos).
    - *Sistemas inteligentes* (pueden aprender nuevos conocimientos por sí solos)
  - Se desarrollan nuevos materiales para superar los problemas de los límites de la miniaturización.
  - Se generaliza y expande el uso de redes de comunicación: Internet, redes inalámbricas, redes de sensores (RFID), etc.

Computadora Mejorada



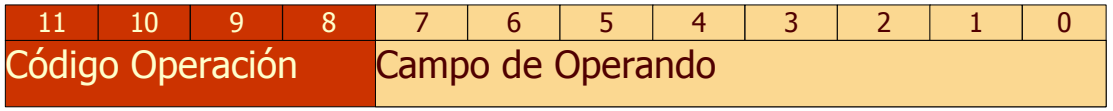
Computadora Mejorada

Componentes y Operaciones de Control

Componente	Operación	Explicación
Memoria	GPR->M	Escribe en contenido de GPR en el lugar direccionado por MAR.
Contador de Programa (PC)	PC+1->PC	Incrementar el contenido de PC
	GPR(AD)->PC	Transmite los 8 bits inferiores de GPR al PC
Registro Direcciones de Memoria (MAR)	PC->MAR	Transmite el contenido del PC al MAR
	GPR(AD)->MAR	Transmite los 8 bits inferiores de GPR al MAR
Registro de Operación (OPR)	GPR(OP)->OPR	Transmite los 4 bits superiores de GPR al OPR
Registro de Propósito General (GPR)	M->GPR	Transmite la palabra direccionada por MAR al GPR
	Acc->GPR	Transmite el contenido de Acc al GPR
	PC->GPR(AD)	Transmite el contenido del PC al GPR, poniendo los 4 bits superiores a 0.
	GPR+1->GPR	Incrementa el GPR
ALU (Acumulador)	GPR+Acc->Acc	Suma el contenido del Acumulador con el contenido del GPR, deja el resultado en Acc
	0->Acc	Limpia Acc
	ROR F,Acc	Desplazamiento cíclico hacia la derecha del Acc junto a F
	ROL F,Acc	Desplazamiento cíclico hacia la izquierda del Acc junto a F
	NOT(Acc)->Acc	Complementa el Acc
	Acc+1->Acc	Incrementa Acc
ALU (Registro F)	0->F	Pone a "0" el registro F
	NOT(F)->F	Complementa F

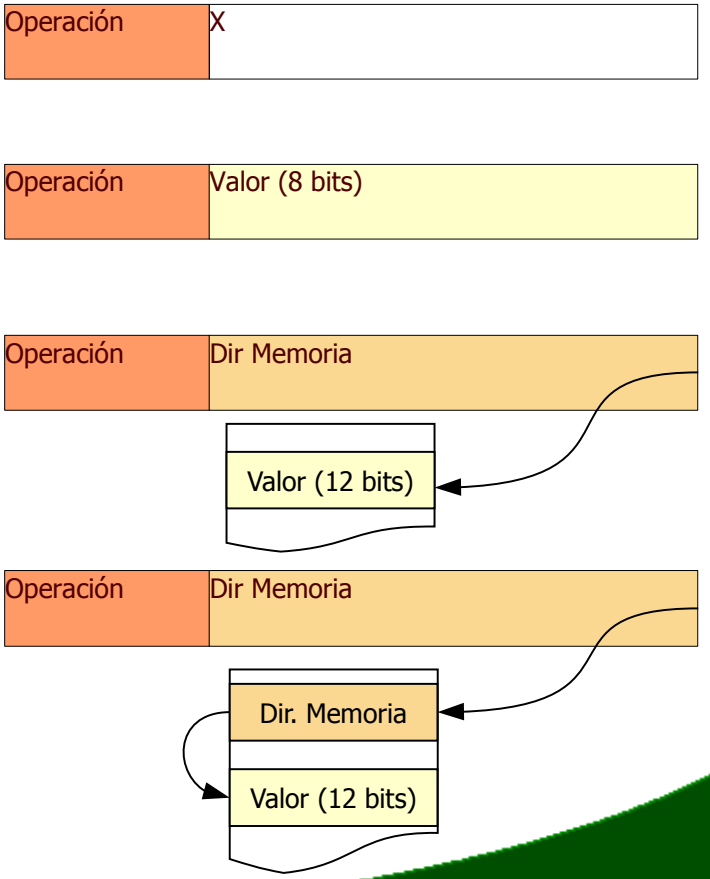
Computadora Mejorada

- Codificación de las Instrucciones
  - Tamaño de las palabras en memoria => 12 bits
  - Repertorio de 16 instrucciones => 4 bits de operación
  - Memoria de 8 bits de direccionamiento => 256 palabras
  - Estructura de codificación:



Computadora Mejorada

- Operandos y Direccionamiento de Memoria:
  - **Modo Implícito**
    - No se indica ningún operando, porque la propia instrucción ya indica sobre qué componente actuará (Ej. CRA)
  - **Modo Inmediato**
    - La instrucción contiene en el campo de operando el valor directo codificado sobre el que se actúa (menor tamaño)
  - **Direccionamiento directo**
    - La instrucción contiene codificada en el campo de operando una posición de memoria, que contiene el valor sobre el que se desea actuar.
  - **Direccionamiento indirecto**
    - La instrucción contiene codificada en el campo de operando una posición de memoria que contiene la dirección de memoria del valor sobre el que se desea operar.

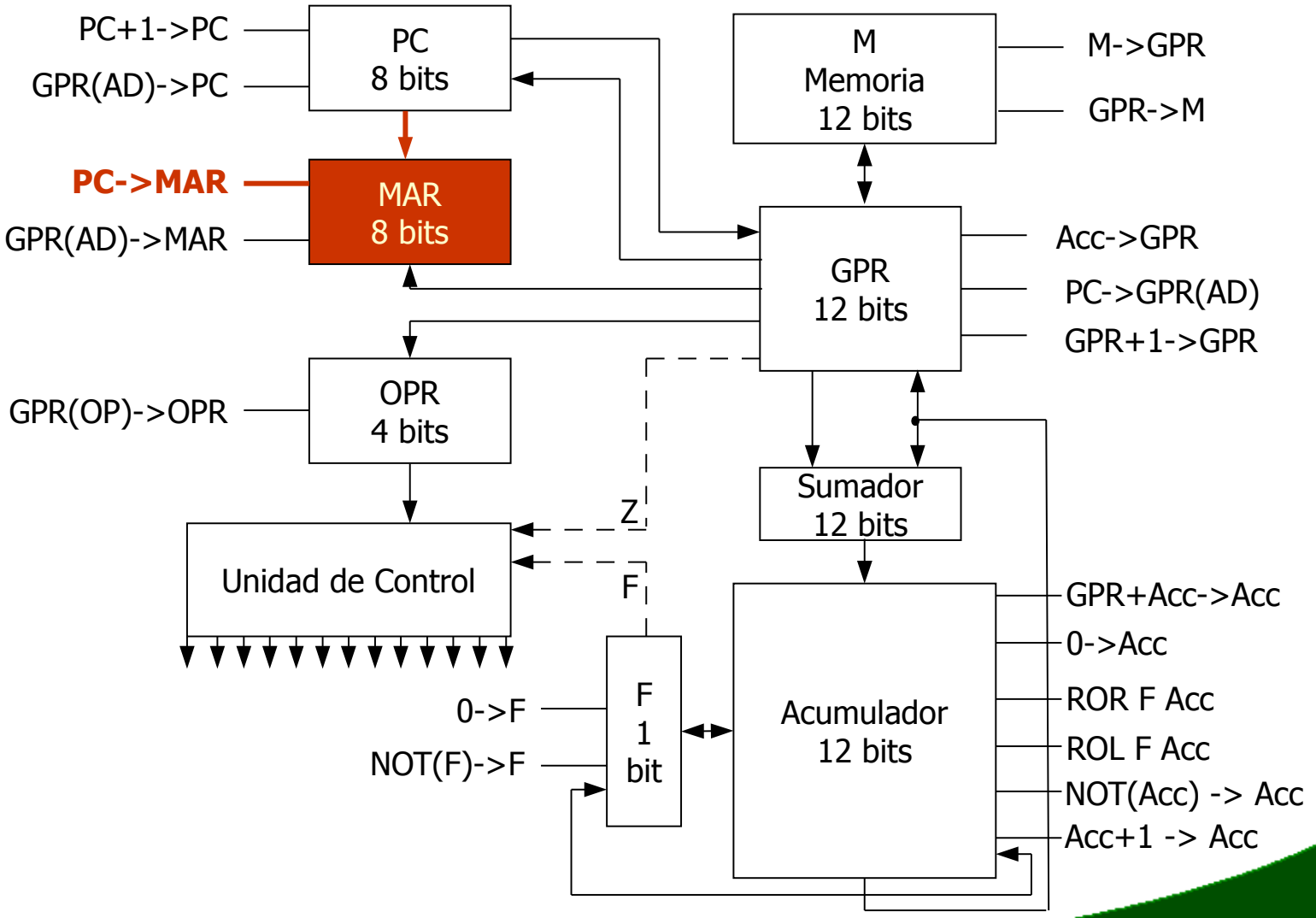


Computadora Mejorada: Ciclo de Búsqueda (FETCH)

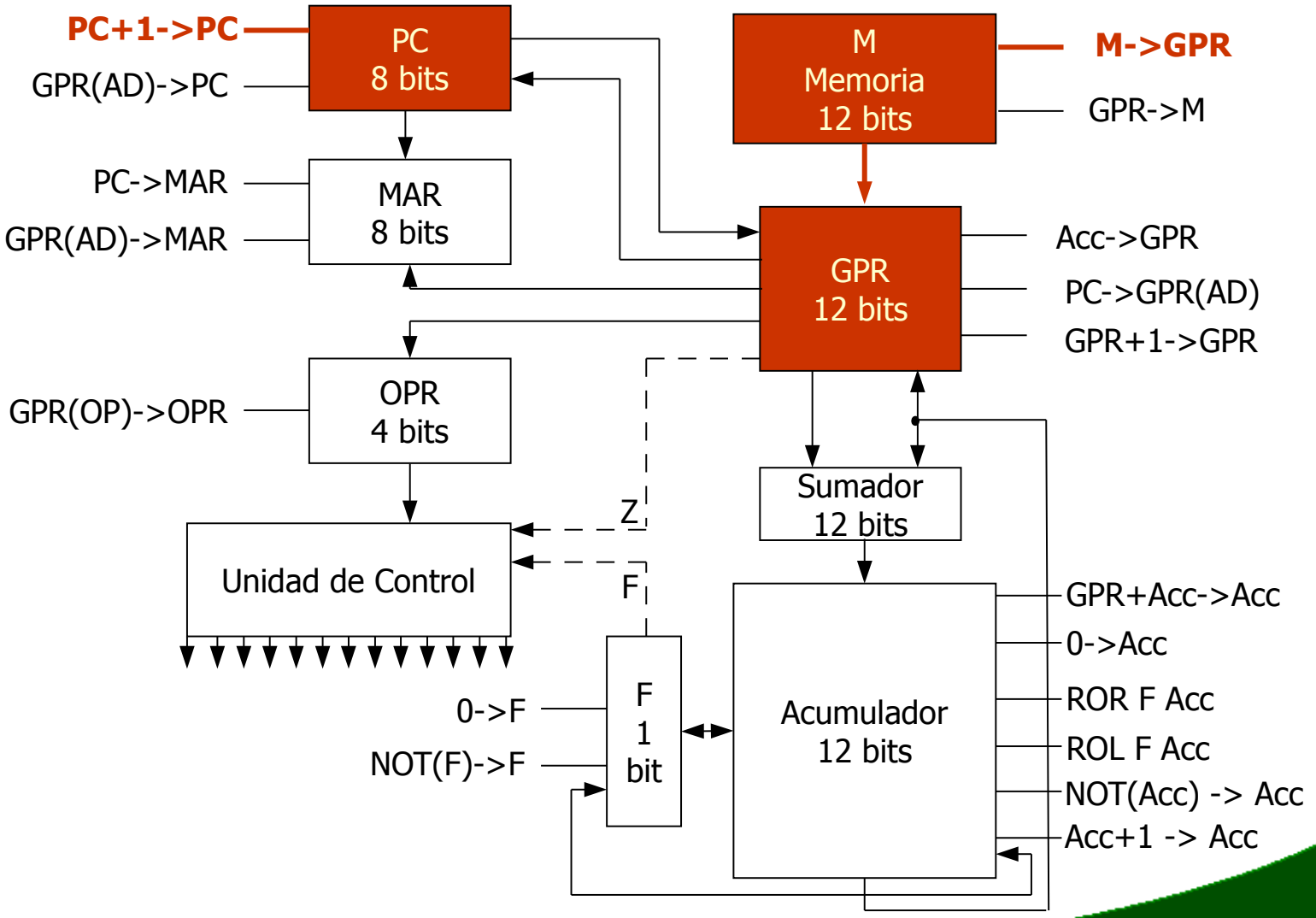
- FETCH
  - Ciclo de búsqueda
  - Captar la siguiente instrucción, decodificarla y preparar el sistema para la captación de la siguiente instrucción, tras el ciclo de ejecución.
  - Secuencia de Microoperaciones:

Ciclo	Operación	Explicación
1	PC->MAR	Transfiere la dirección de memoria de la siguiente instrucción (PC) al MAR.
2	M->GPR PC+1->PC	Leo el contenido de la memoria al que apunta MAR (que contiene la codificación de la siguiente instrucción). Incremento el PC para apuntar a la siguiente instrucción.
3	GPR(OP)->OPR	Transfiero los 4 bits superiores de GPR (que contienen el campo de operación) al registro OPR para su decodificación

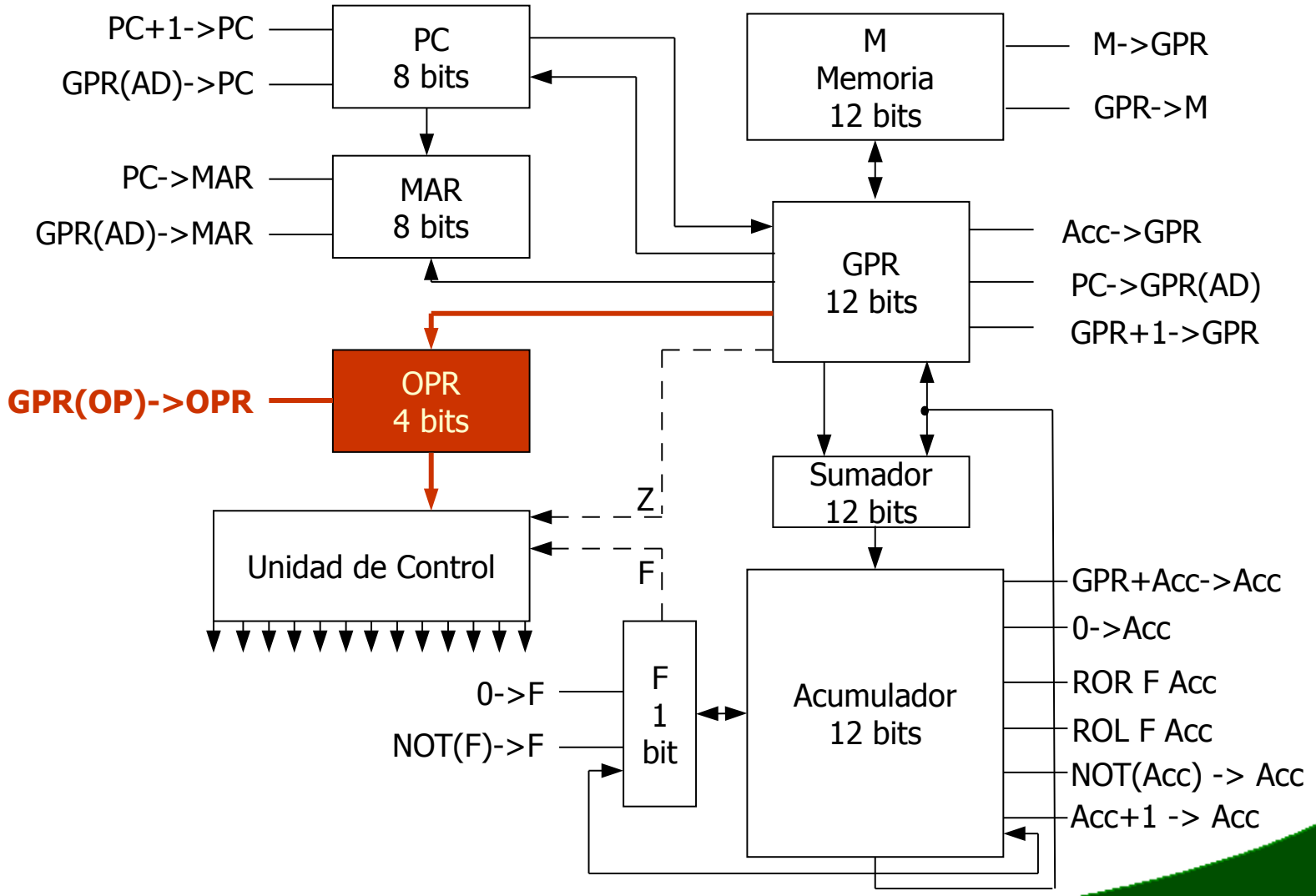
Computadora Mejorada: Ciclo de Búsqueda



Computadora Mejorada: Ciclo de Búsqueda



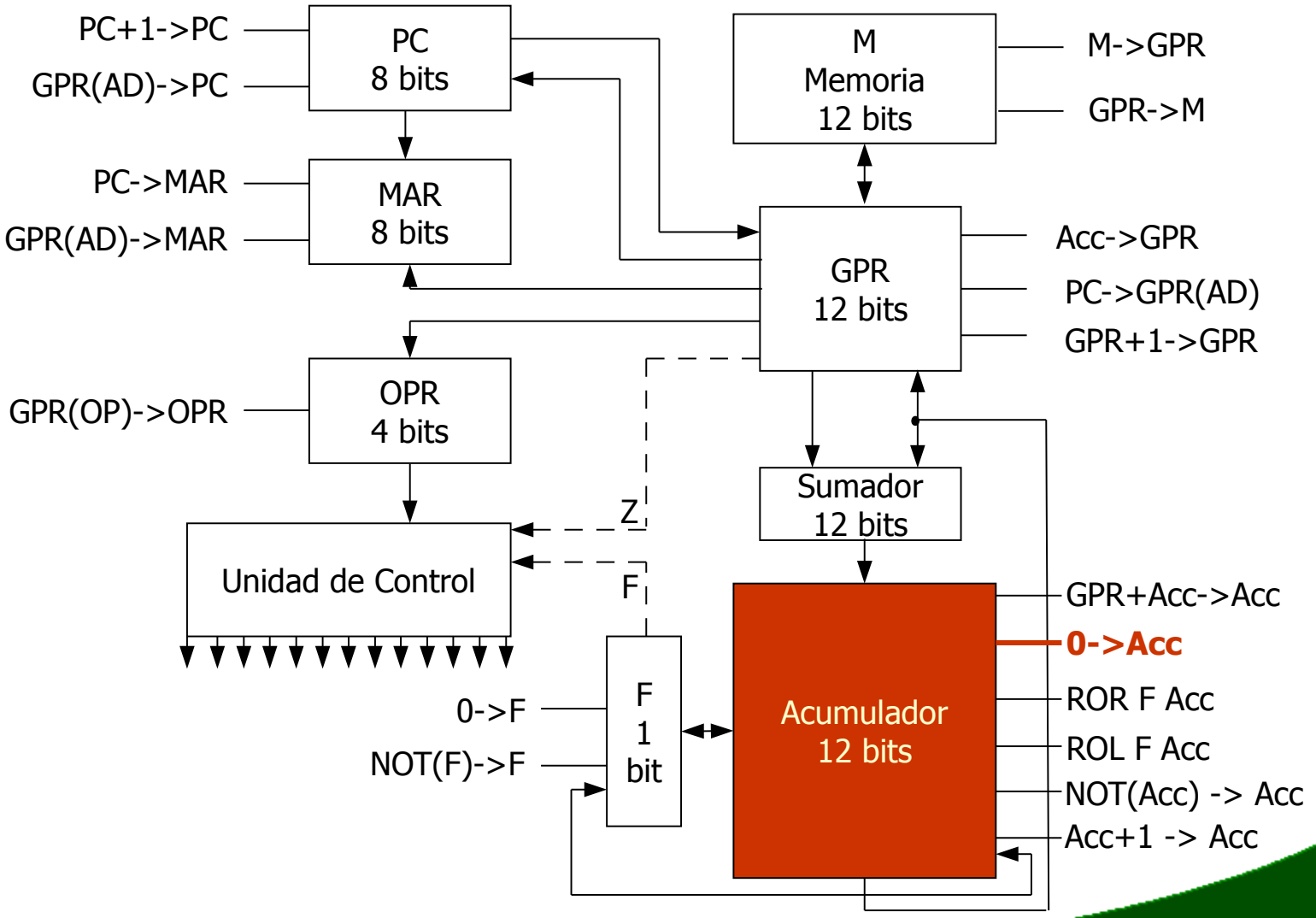
Computadora Mejorada: Ciclo de Búsqueda



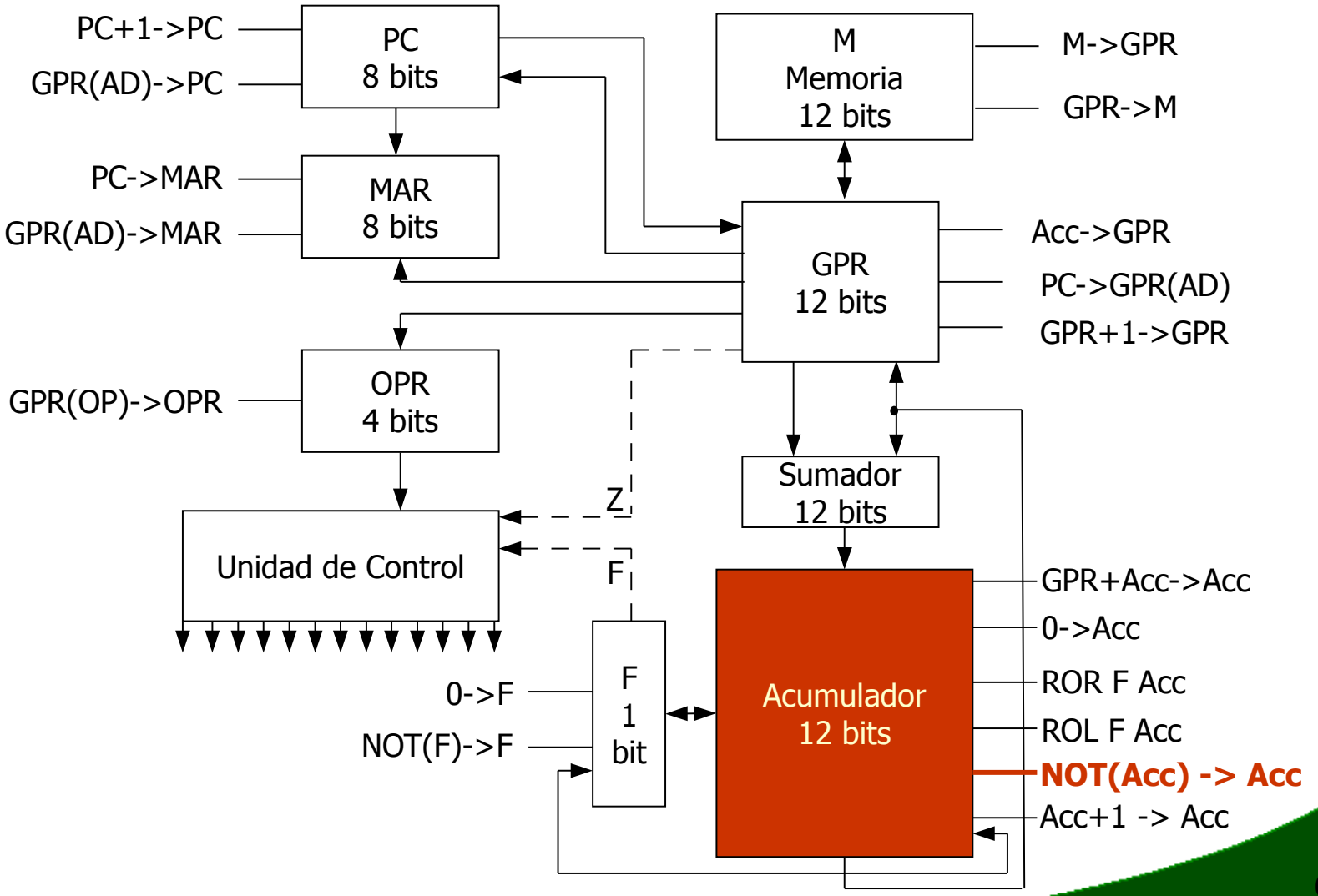
Computadora Mejorada: Instrucciones

- Instrucciones Monociclo sin Operando
  - Instrucciones que coinciden con microoperaciones:
    - CRA (*CleaR Accumulator*) : 0->Acc
    - CTA (*ComplementT Accumulator*) : NOT(Acc) -> Acc
    - ITA (*IncrementT Accumulator*) : Acc+1 -> Acc
    - ROR (*ROtation Right*) : ROR F,Acc
    - ROL (*ROtation Left*) : ROL F,Acc
    - CRF (*CleaR "F"*) : 0->F
    - CTF (*ComplementT "F"*) : NOT(F) ->F
    - SFZ (*Skip next instruction if "F" is Zero*) : PC+1->PC, si F=0

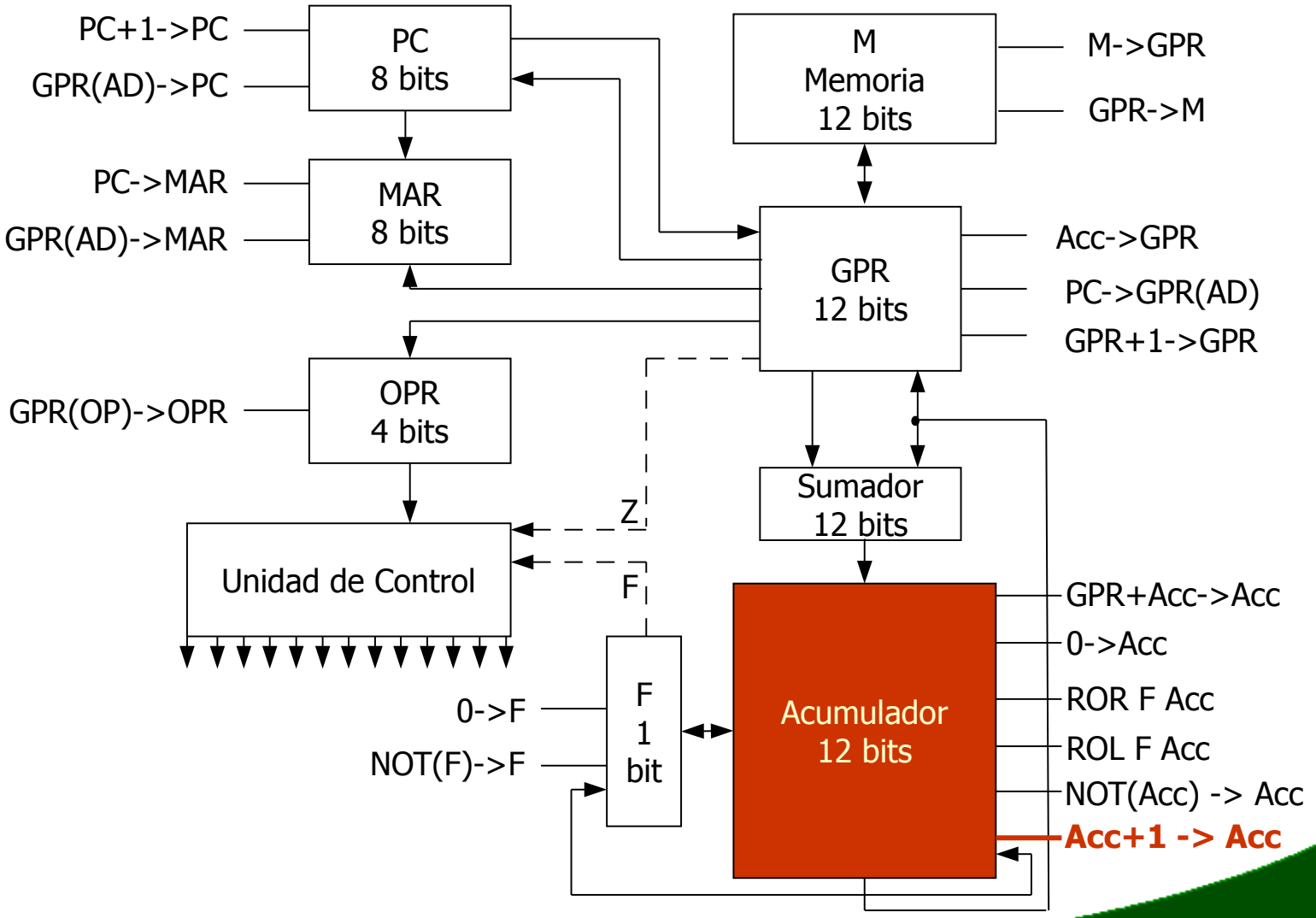
Computadora Mejorada: CRA



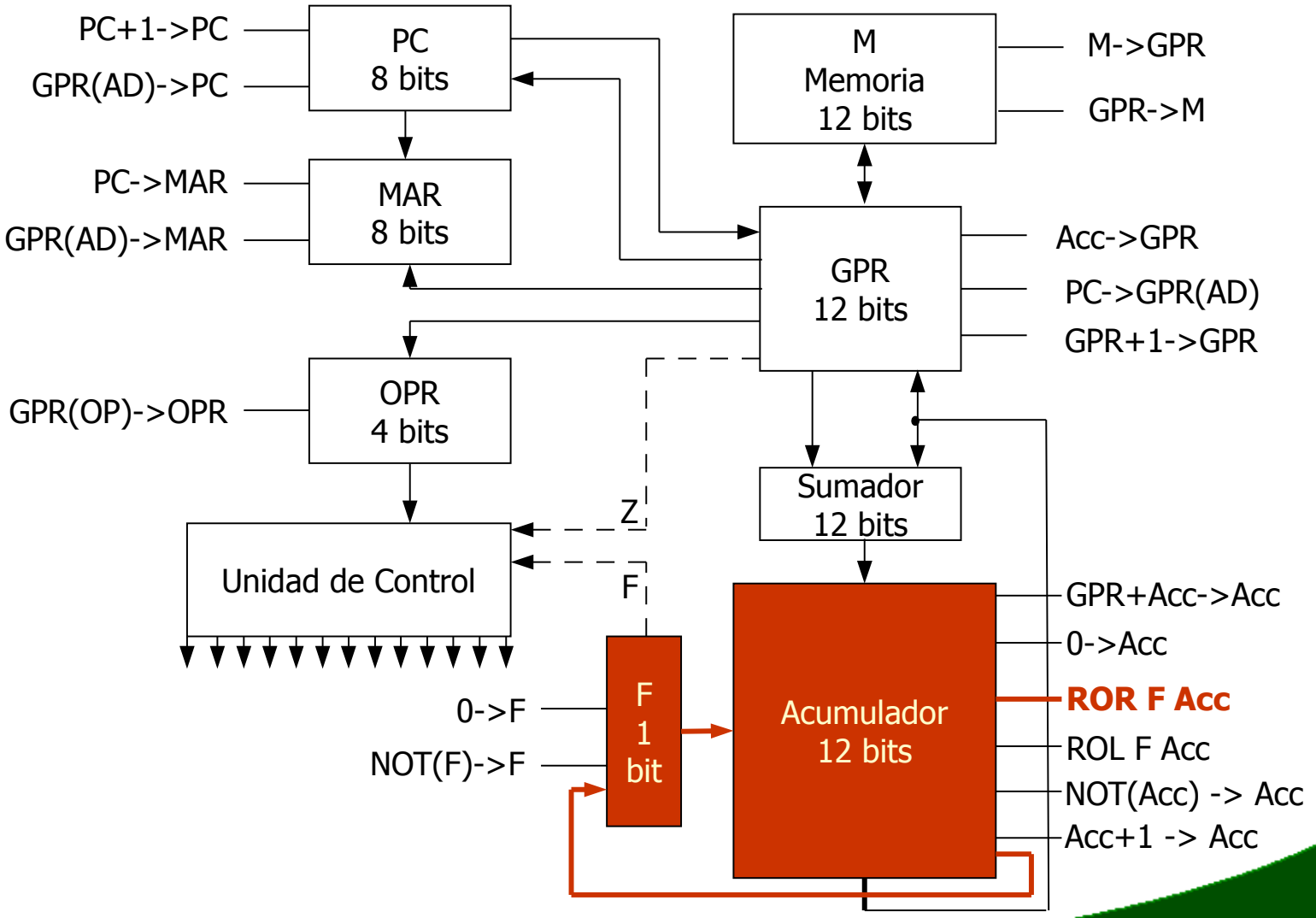
Computadora Mejorada: CTA



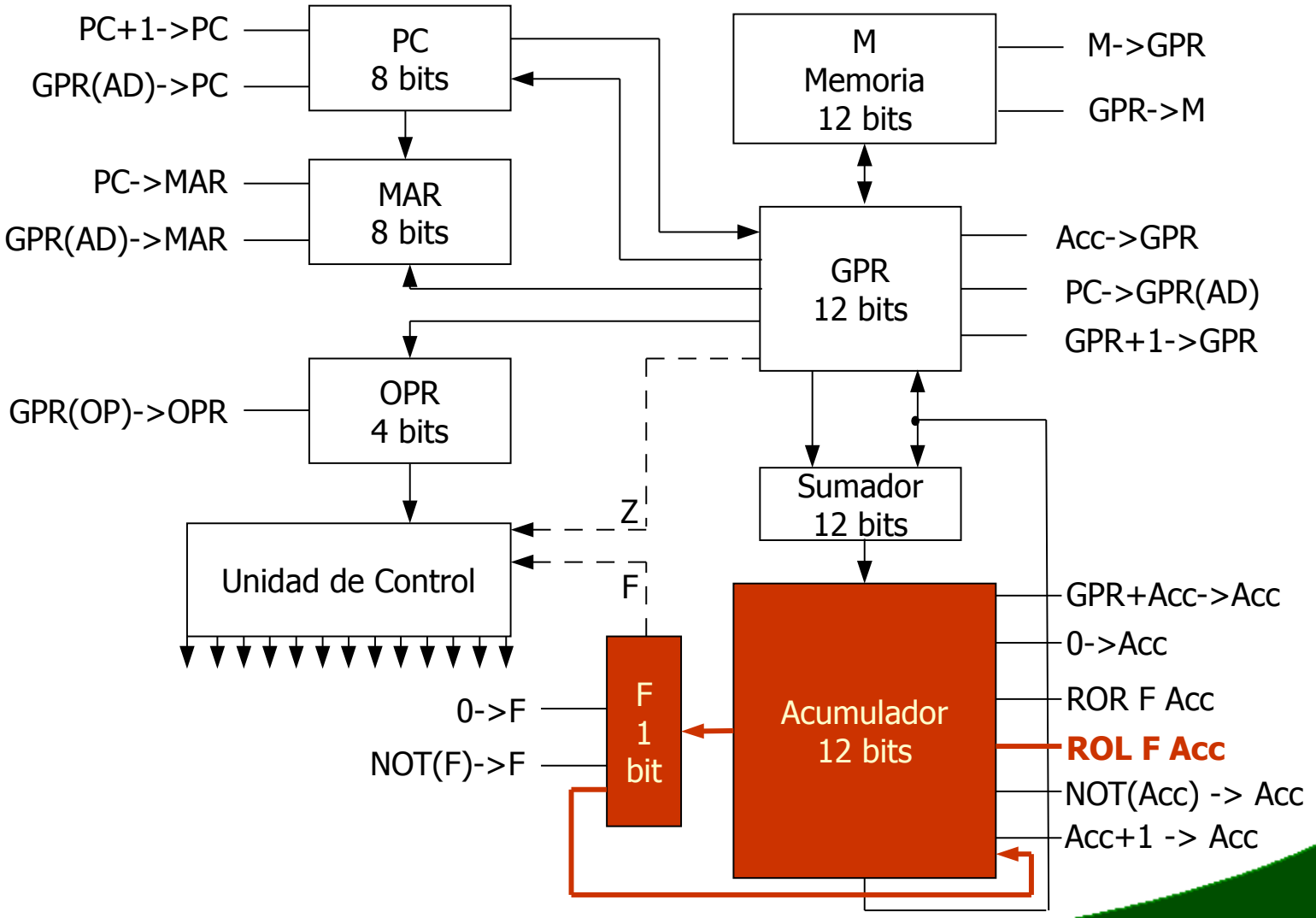
Computadora Mejorada: ITA



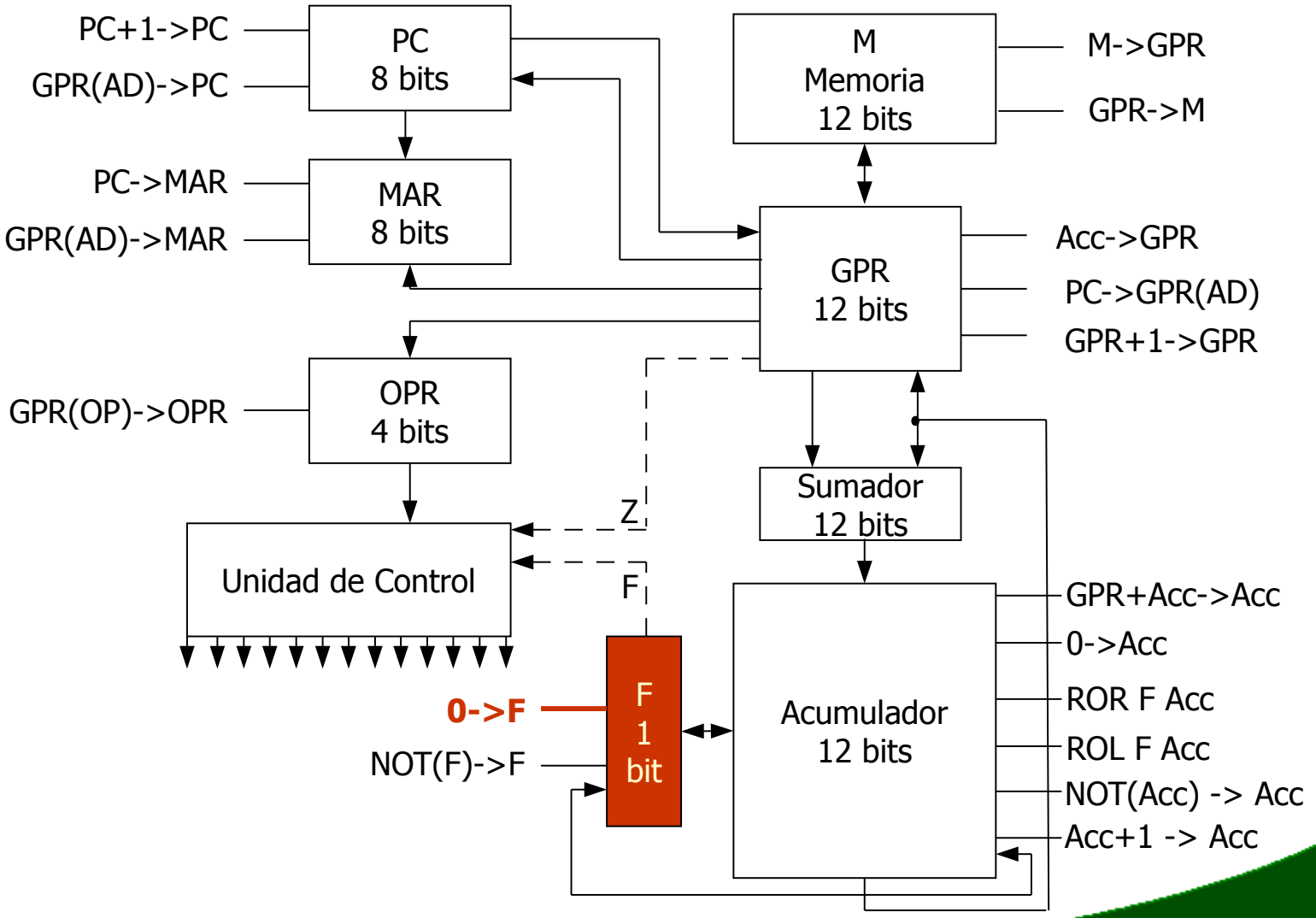
Computadora Mejorada: ROR



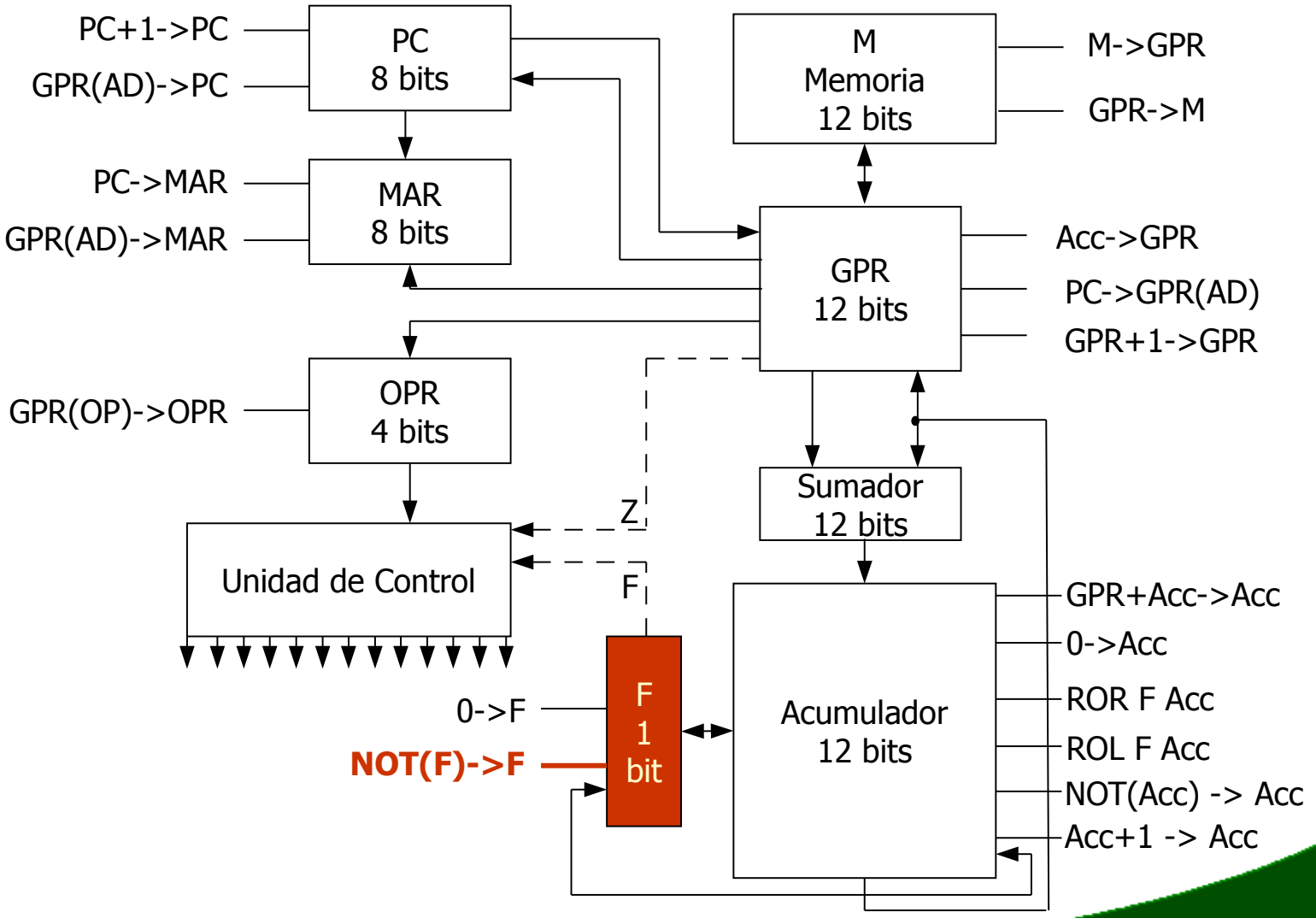
Computadora Mejorada: ROL



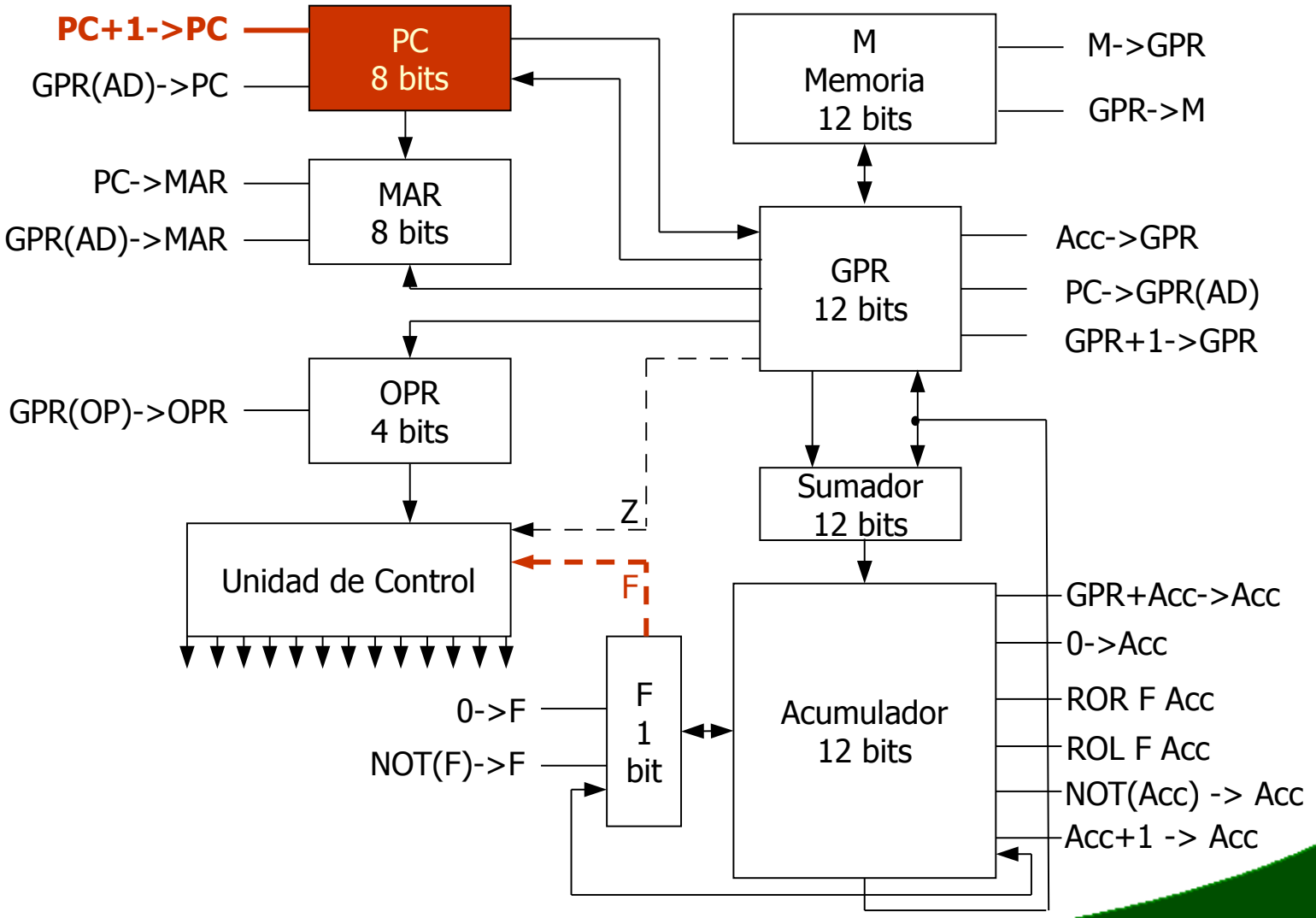
Computadora Mejorada: CRF



Computadora Mejorada: CTF



Computadora Mejorada: SFZ



Computadora Mejorada

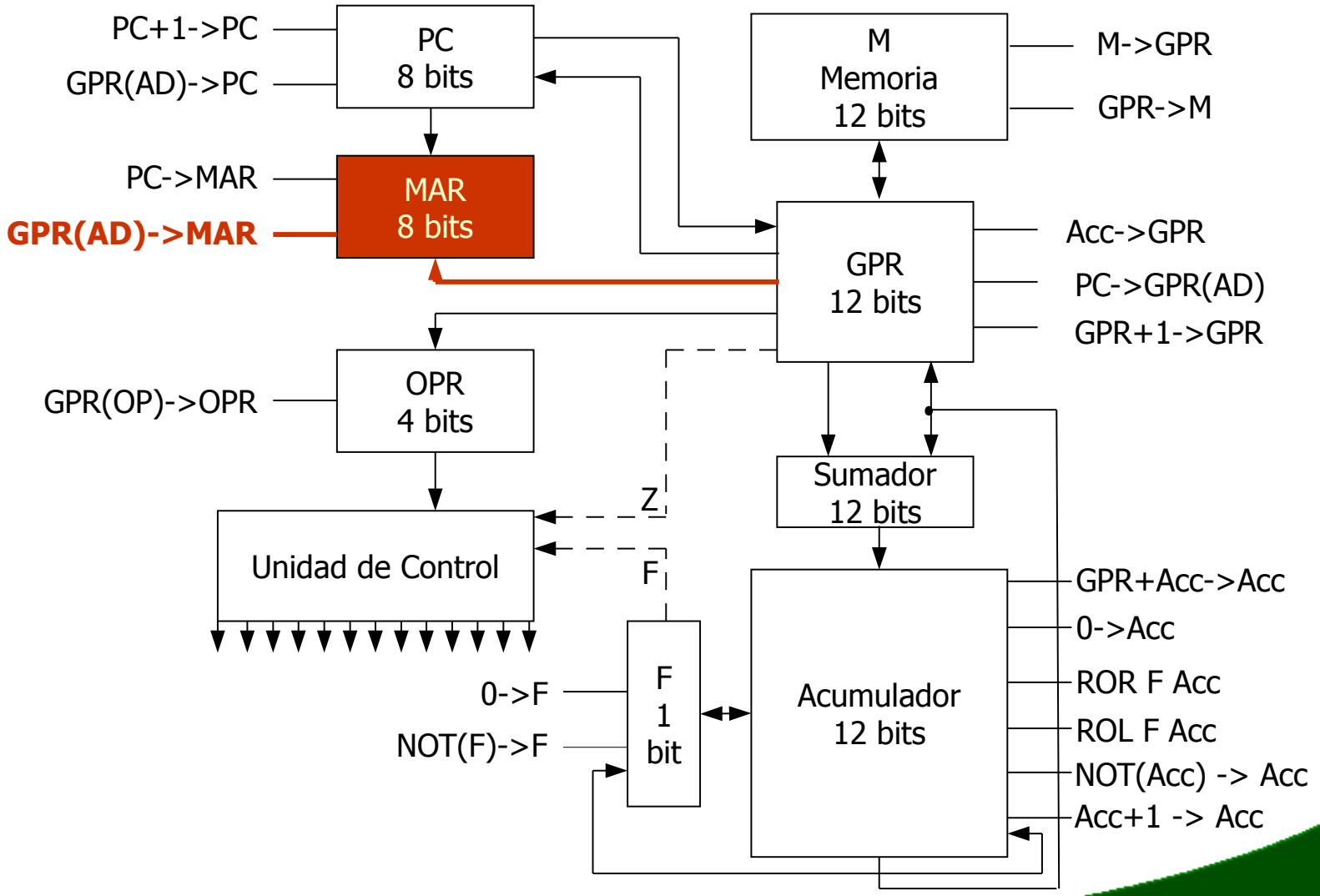
- Instrucciones con Operando:
  - Instrucciones Aritméticas:
    - ADD d (*ADDition*)
    - ADDI d (*ADDition Indirect*)
  - Instrucciones de Transferencia:
    - STA d (*STore Accumulator*)
  - Instrucciones de Control de Flujo:
    - JMP d (*JuMP*)
    - CSR d (*Call SubRutine*)
    - JMPI d (*JuMP Indirect*)
  - Instrucciones Mixtas (Aritmética/Transferencia/Control de Flujo):
    - ISZ d (*Increment memory and Skip next instruction if Zero*)
- Otras Instrucciones
  - HLT (*HaLT*)

Computadora Mejorada: ADD d

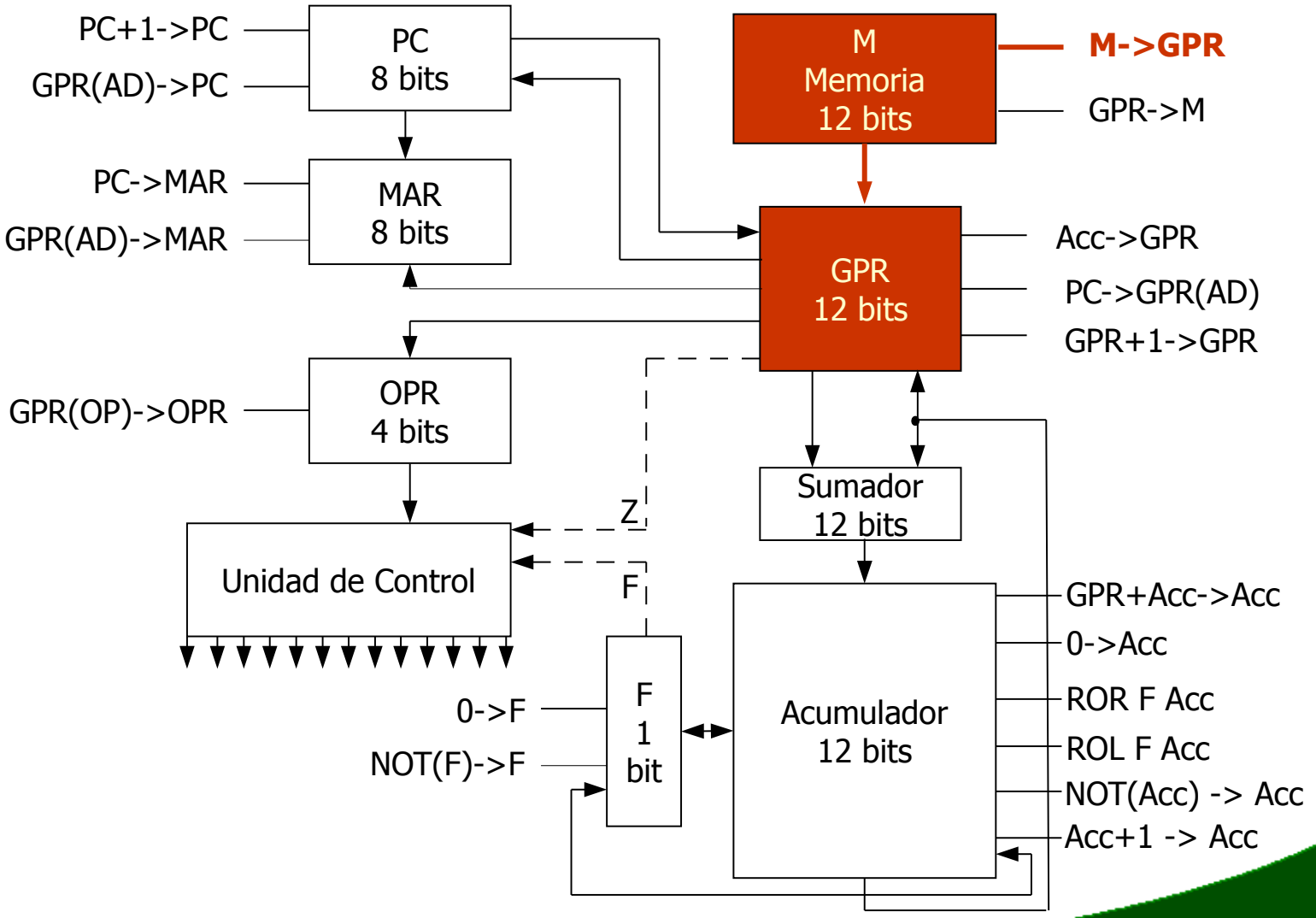
- ADD dirección
  - Suma Directa con el Acumulador
  - Suma el contenido actual del acumulador con el valor que se encuentra en la dirección de memoria especificada en el campo de operando.
  - Secuencia de Microoperaciones:

Ciclo	Operación	Explicación
1	GPR(AD)->MAR	Transfiere la dirección de memoria donde está el valor desde el campo de operando (que se encuentra en los 8 bits inferiores del GPR) al MAR.
2	M->GPR	Lee desde la memoria la dirección que se encontraba en el campo de operando y el valor se deposita en GPR
3	GPR+Acc->Acc	Sumamos el contenido de GPR (nuevo valor) con el contenido de Acc y lo dejamos en Acc.

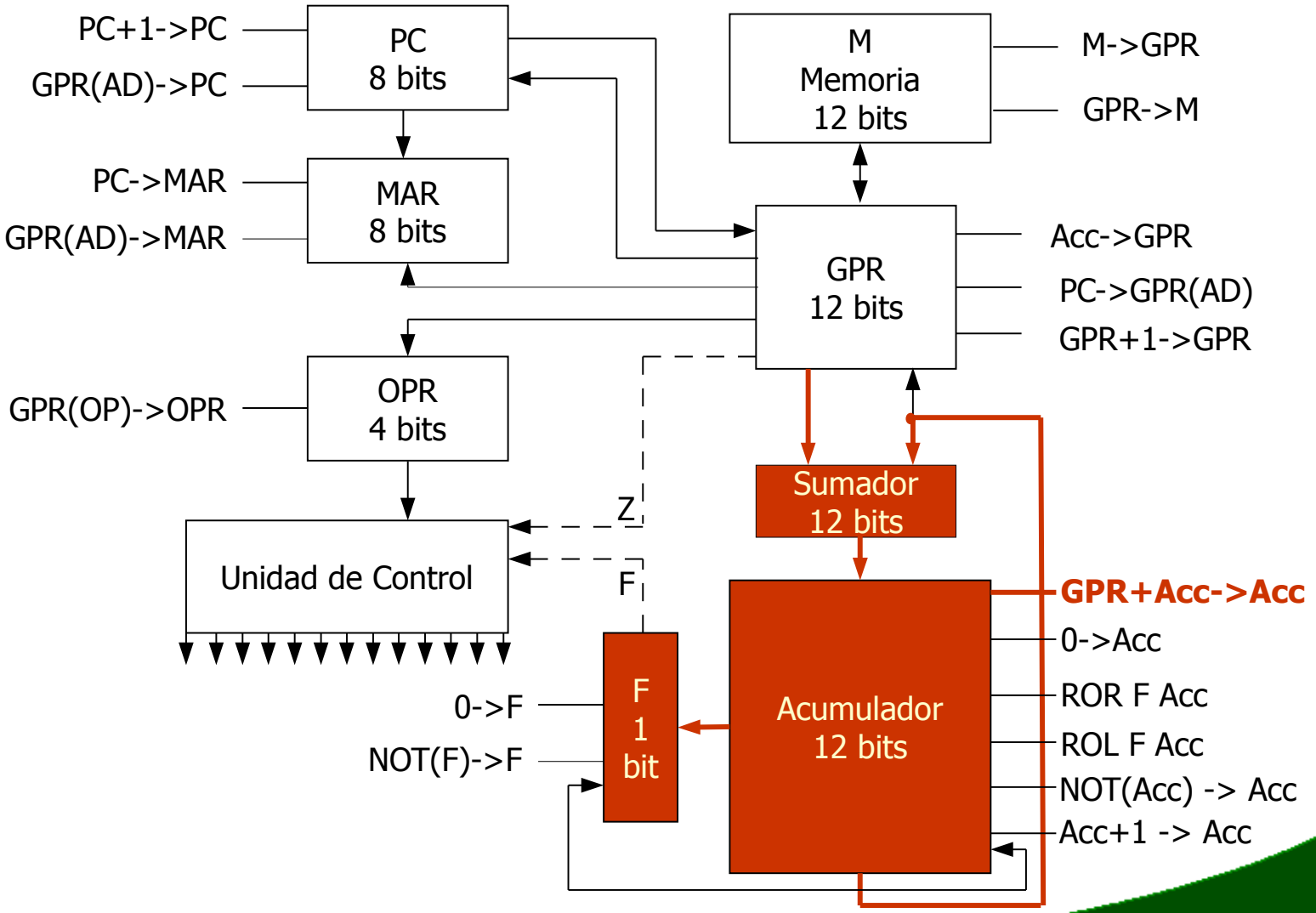
Computadora Mejorada: ADD d



Computadora Mejorada: ADD d



Computadora Mejorada: ADD d

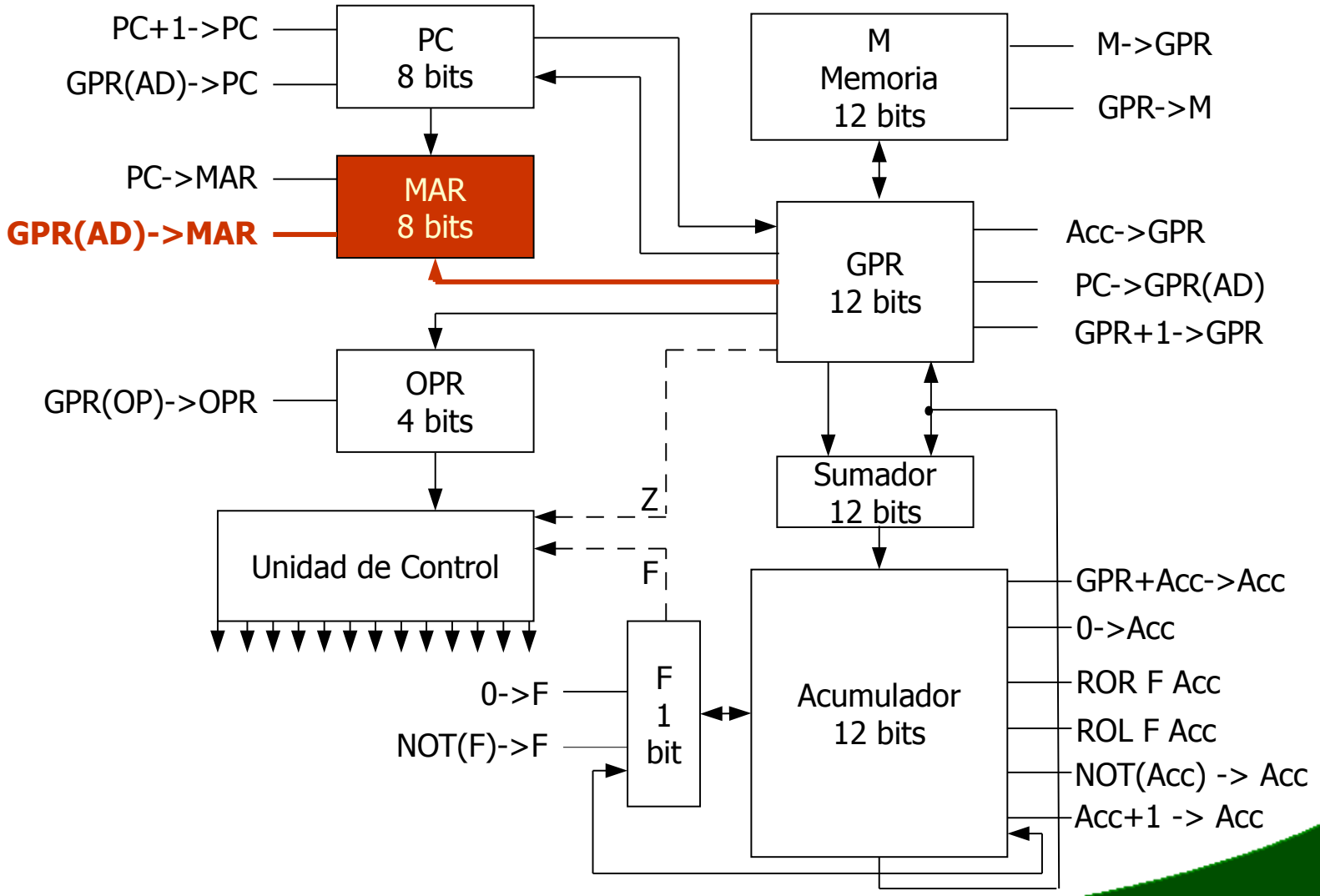


Computadora Mejorada: ADDI d

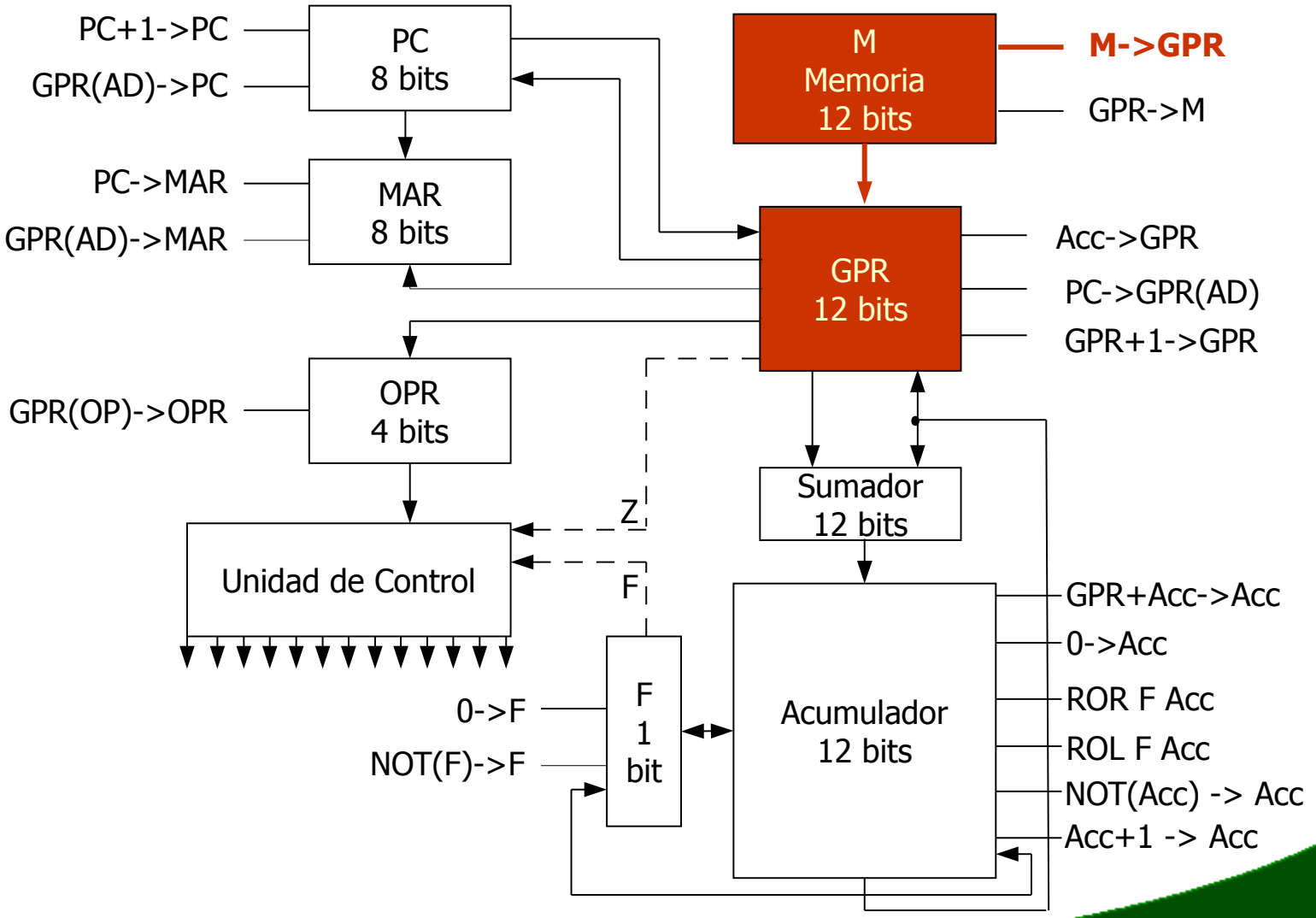
- ADDI dirección
  - Suma Indirecta con el Acumulador
  - Suma el contenido actual del acumulador con el valor que se encuentra en la dirección de memoria especificada en el campo de operando.
  - Secuencia de Microoperaciones:

Ciclo	Operación	Explicación
1	GPR(AD)->MAR	Transfiere la dirección de memoria donde está la dirección de memoria que almacena el valor desde el campo de operando (que se encuentra en los 8 bits inferiores del GPR) al MAR.
2	M->GPR	Lee desde la memoria la dirección que se encontraba en el campo de operando y la dirección de memoria se deposita en GPR
3	GPR(AD)->MAR	Transfiere la nueva dirección de memoria donde está el valor desde el campo de operando (que se encuentra en los 8 bits inferiores del GPR) al MAR. (Los 4 bits superiores son descartables)
4	M->GPR	Lee desde la memoria la dirección que se encontraba en el campo de operando y el valor se deposita en GPR
5	GPR+Acc->Acc	Sumamos el contenido de GPR (nuevo valor) con el contenido de Acc y lo dejamos en Acc.

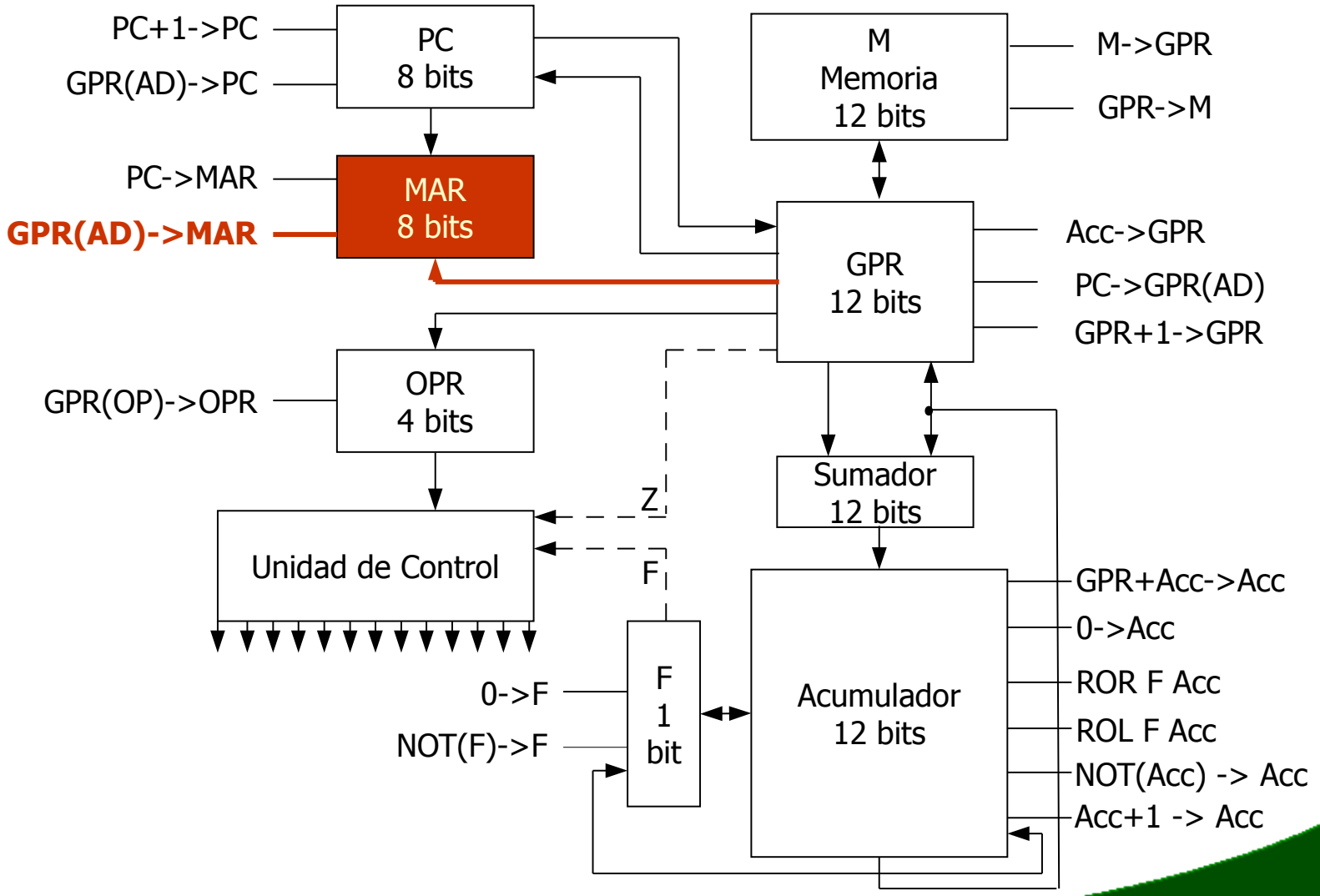
Computadora Mejorada: ADDI d



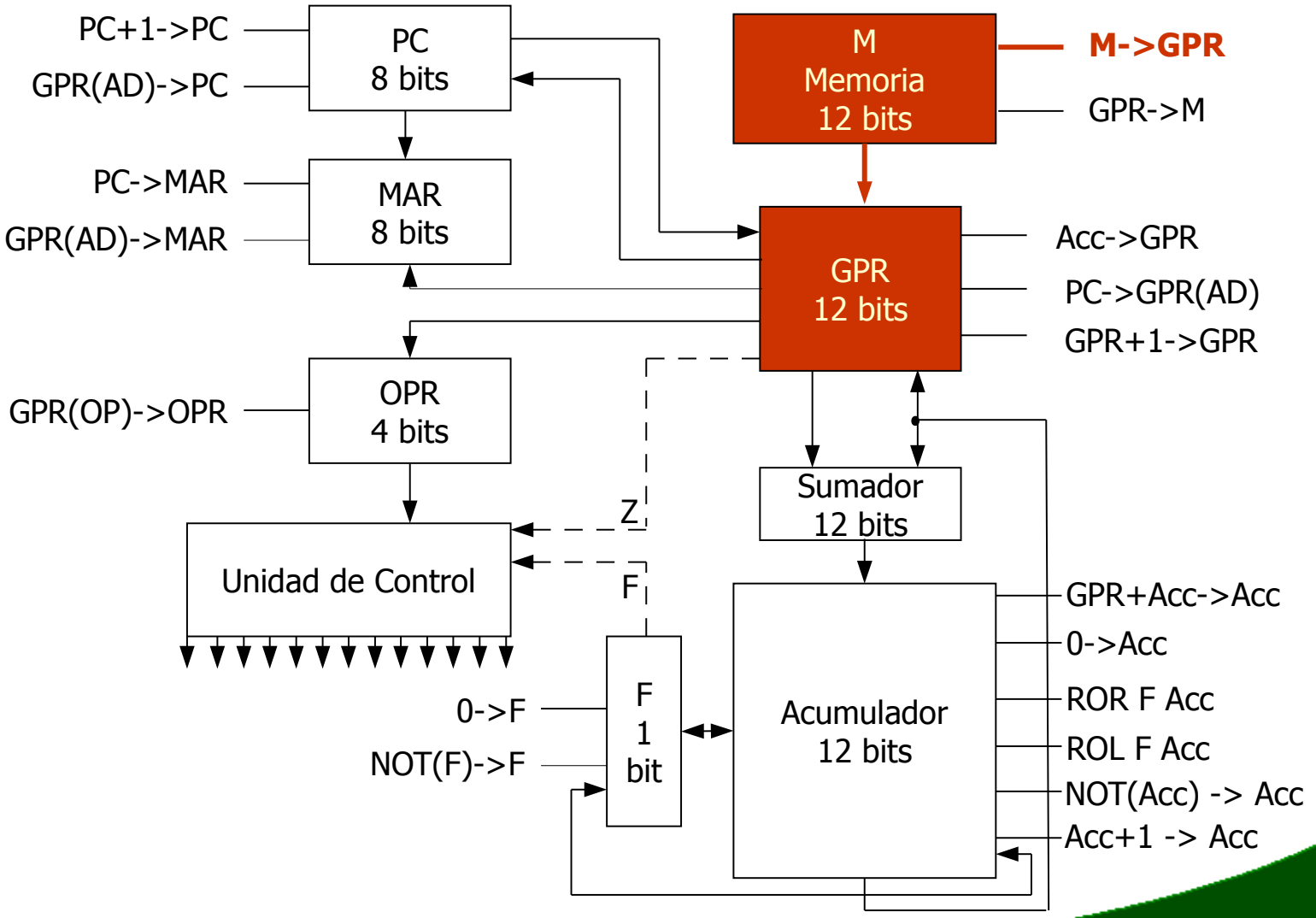
Computadora Mejorada: ADDI d



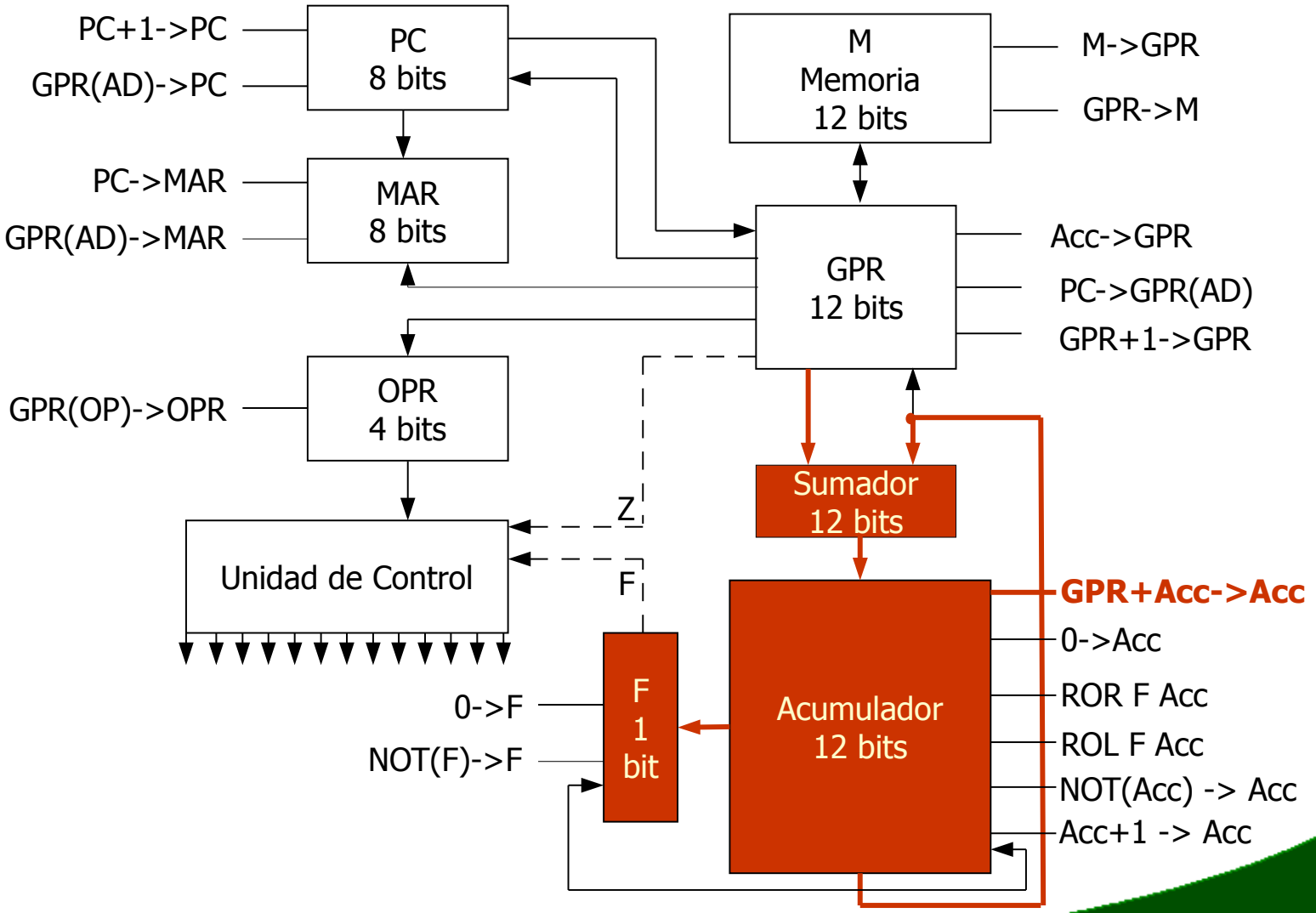
Computadora Mejorada: ADDI d



Computadora Mejorada: ADDI d



Computadora Mejorada: ADDI d

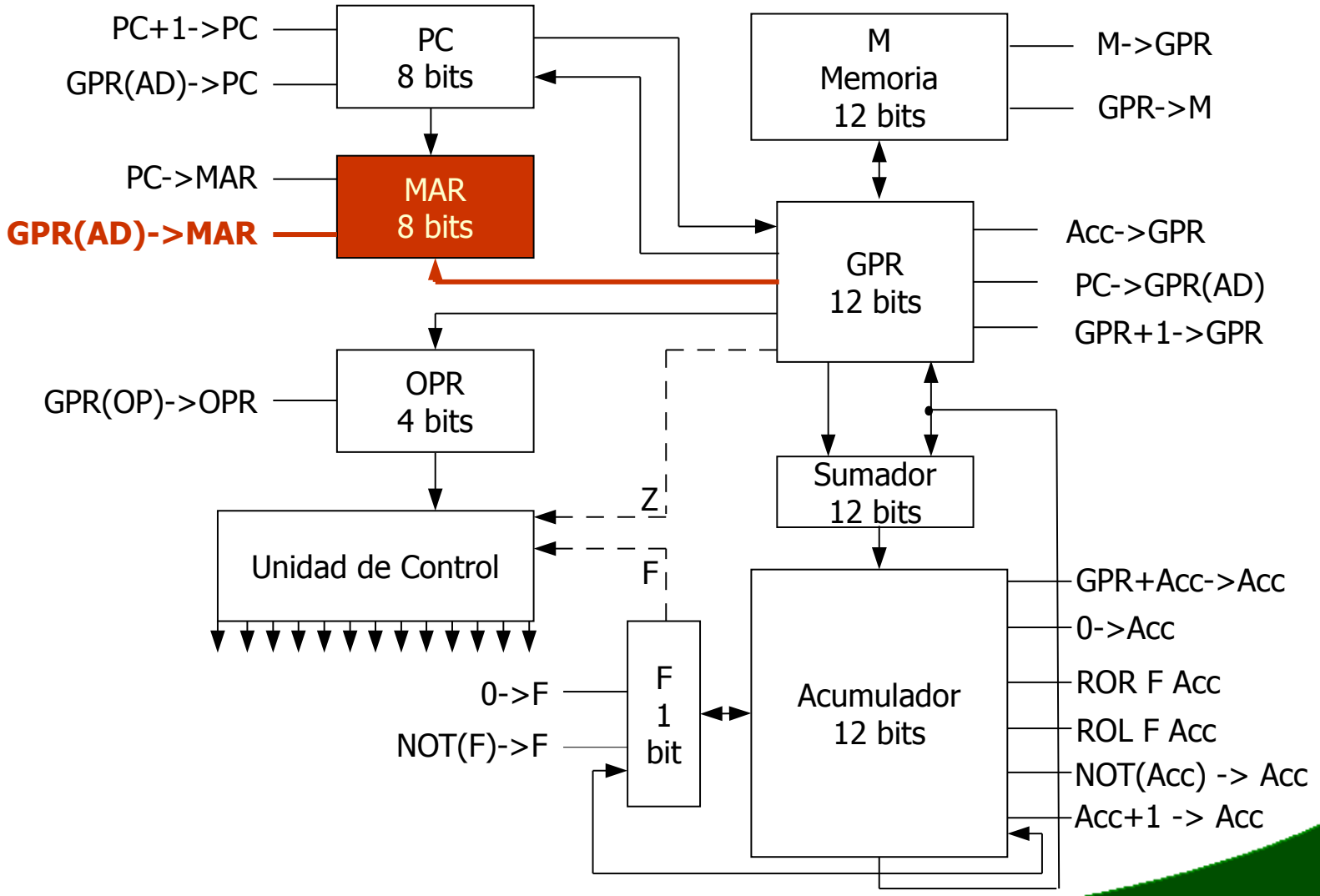


Computadora Mejorada: STA d

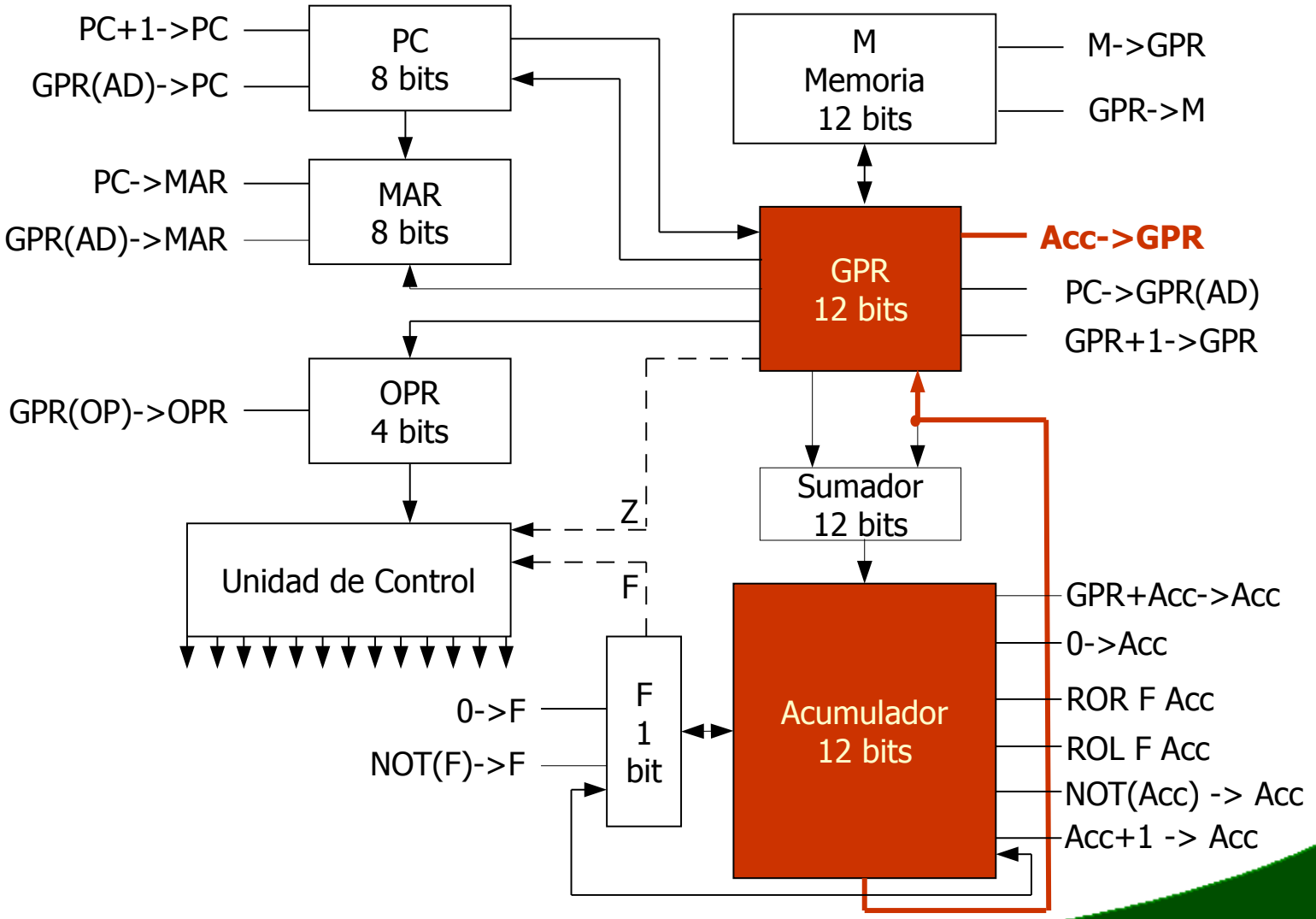
- STA dirección
  - Almacenamiento Directo del Acumulador en Memoria
  - Almacena el contenido actual del acumulador en la dirección de memoria especificada en el campo de operando.
  - Secuencia de Microoperaciones:

Ciclo	Operación	Explicación
1	GPR(AD)->MAR	Transfiere la dirección de memoria donde está el valor desde el campo de operando (que se encuentra en los 8 bits inferiores del GPR) al MAR.
2	Acc->GPR	Copia el valor almacenado en Acc al registro GPR
3	GPR->M	Almacenamos el valor que hay en el GPR en la posición de memoria que se indica en MAR.

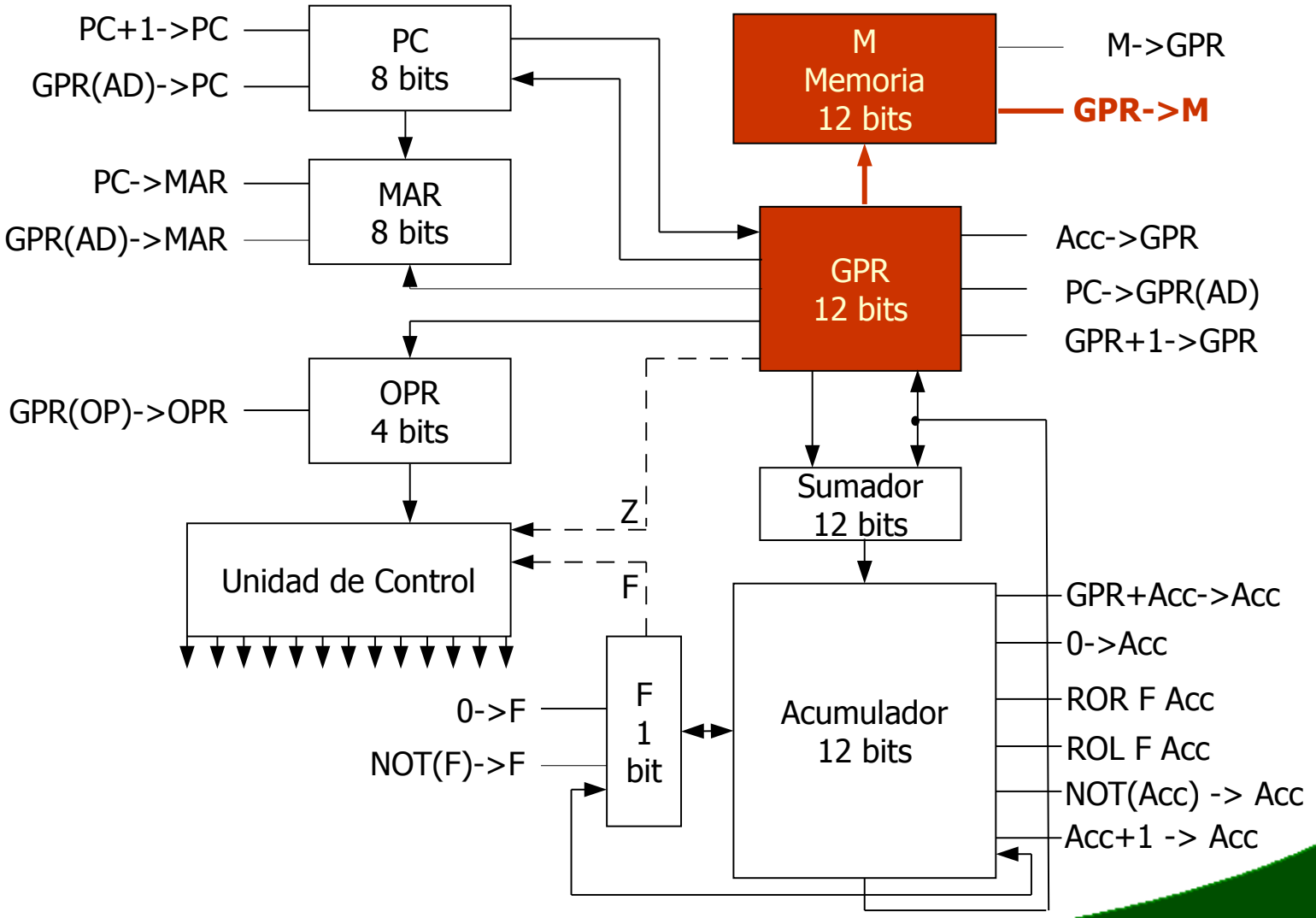
Computadora Mejorada: STA d



Computadora Mejorada: STA d



Computadora Mejorada: STA d

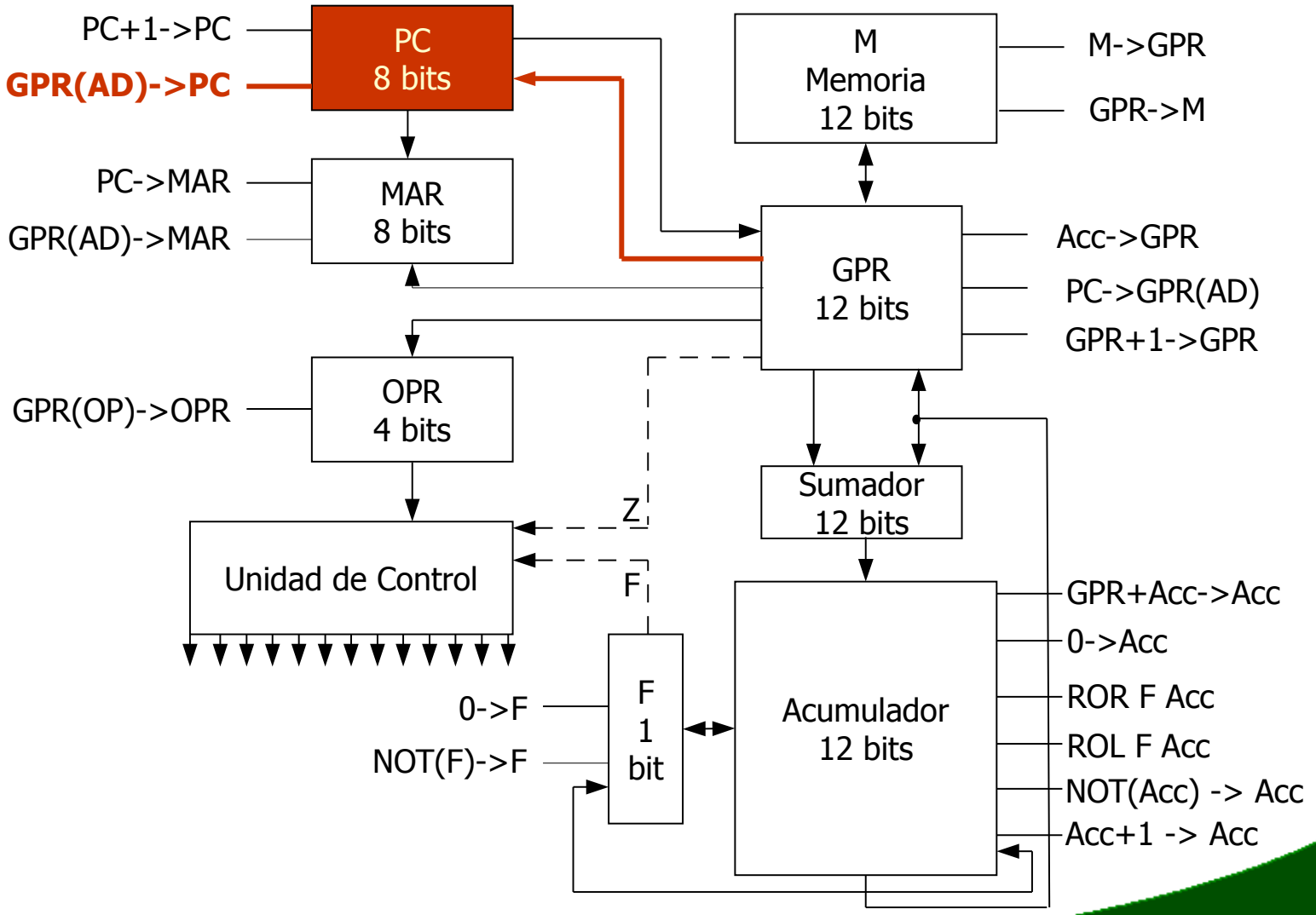


Computadora Mejorada: JMP d

- JMP dirección
  - Salto directo en el flujo de ejecución del programa
  - La siguiente instrucción que se ejecutará es la que se encuentra en la dirección de memoria especificada en el campo de operando.
  - Secuencia de Microoperaciones:

Ciclo	Operación	Explicación
1	GPR(AD)->PC	Transferir la posición de memoria indicada en los 8 bits menos significativos de GPR a PC.

Computadora Mejorada: JMP d

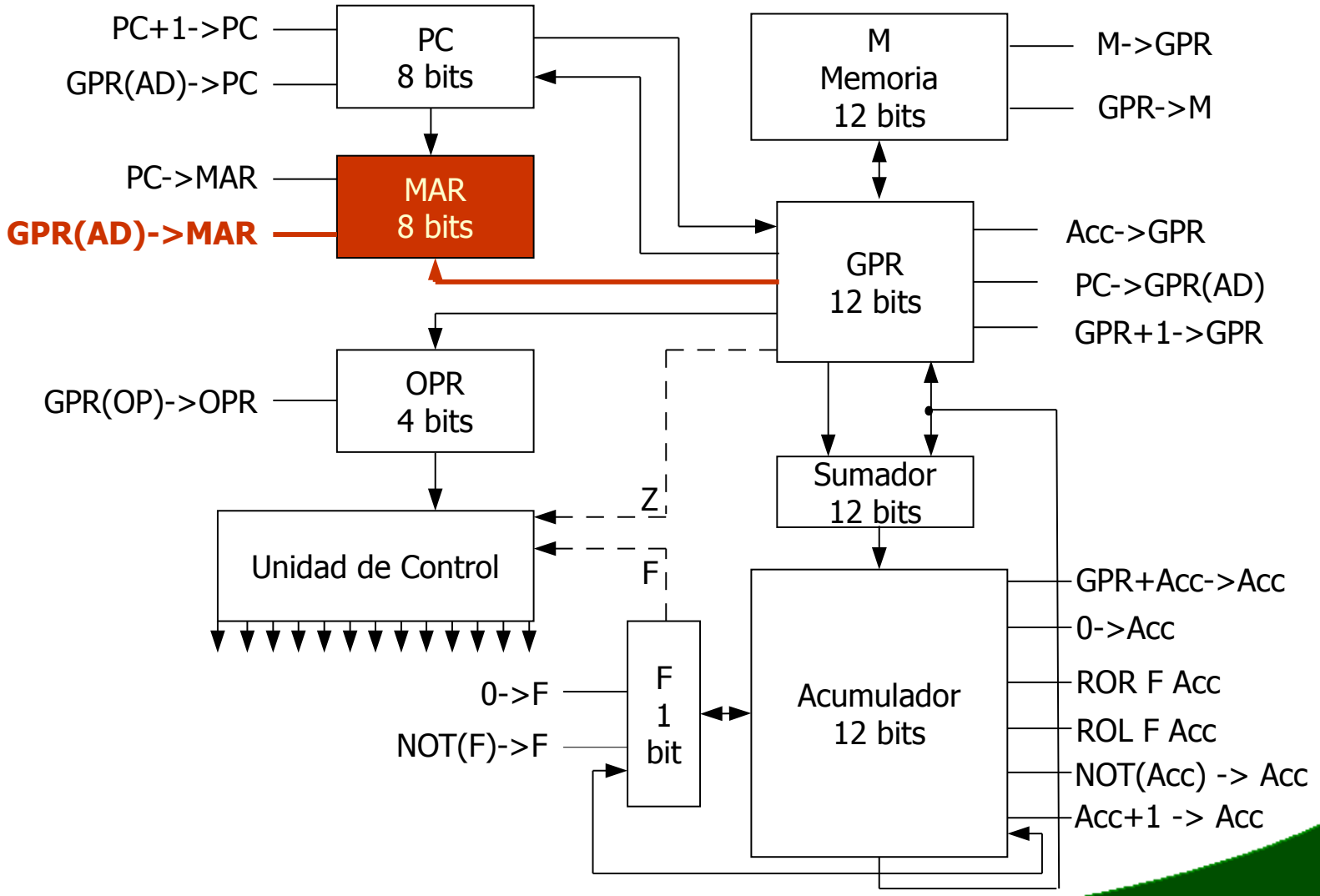


Computadora Mejorada: JMPI d

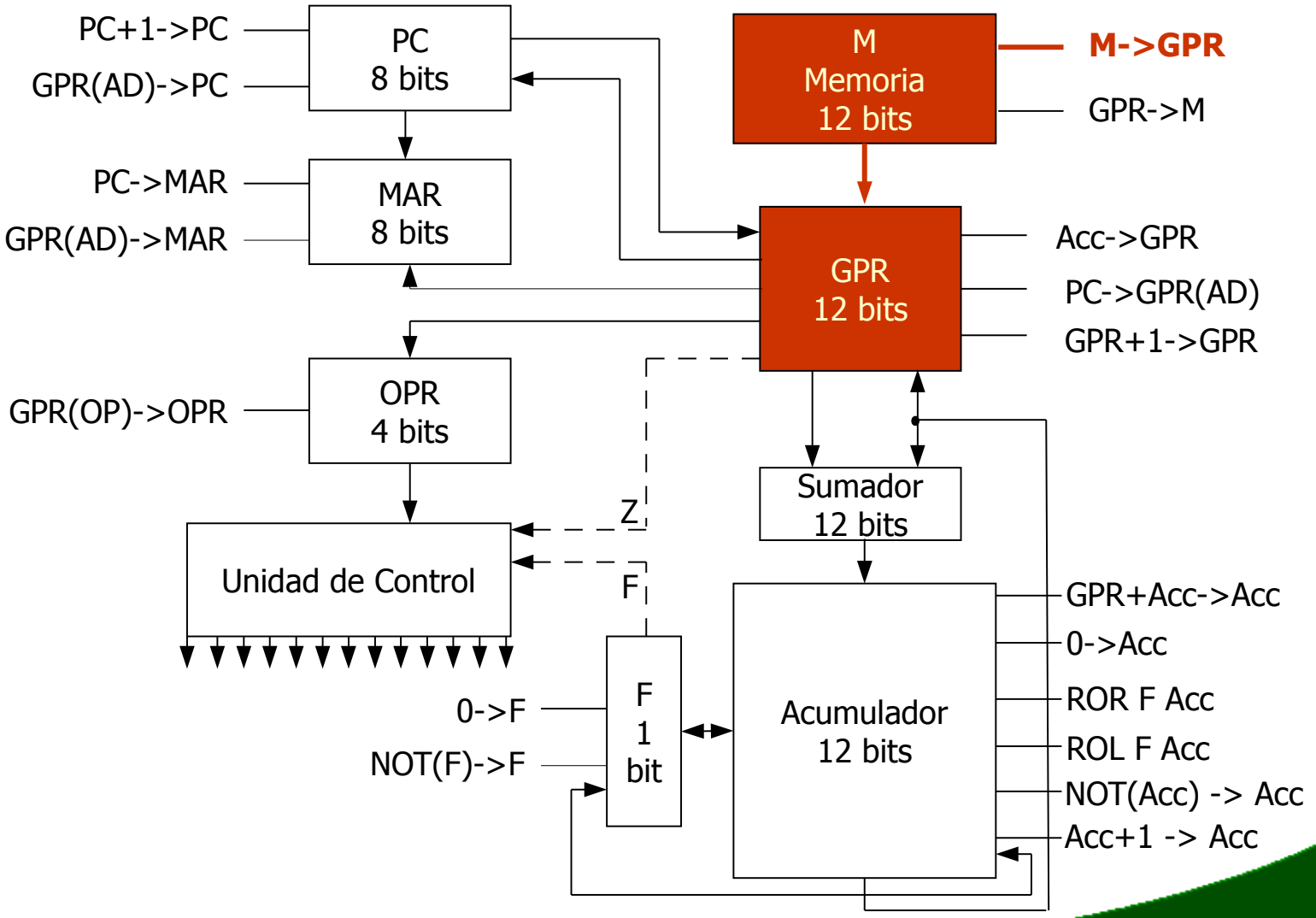
- JMPI dirección
  - Salto indirecto en el flujo de ejecución del programa
  - La siguiente instrucción que se ejecutará es la que se encuentra en la dirección de memoria cuya dirección de memoria viene especificada en el campo de operando.
  - Secuencia de Microoperaciones:

Ciclo	Operación	Explicación
1	GPR(AD)->MAR	Transmitir la dirección indicada en el campo de operando al MAR.
2	M->GPR(AD)	Leer el contenido de memoria al que apunta el MAR (que contiene la posición de memoria de la siguiente instrucción que se va a ejecutar)
3	GPR(AD)->PC	Transferir la posición de memoria indicada en los 8 bits menos significativos de GPR a PC.

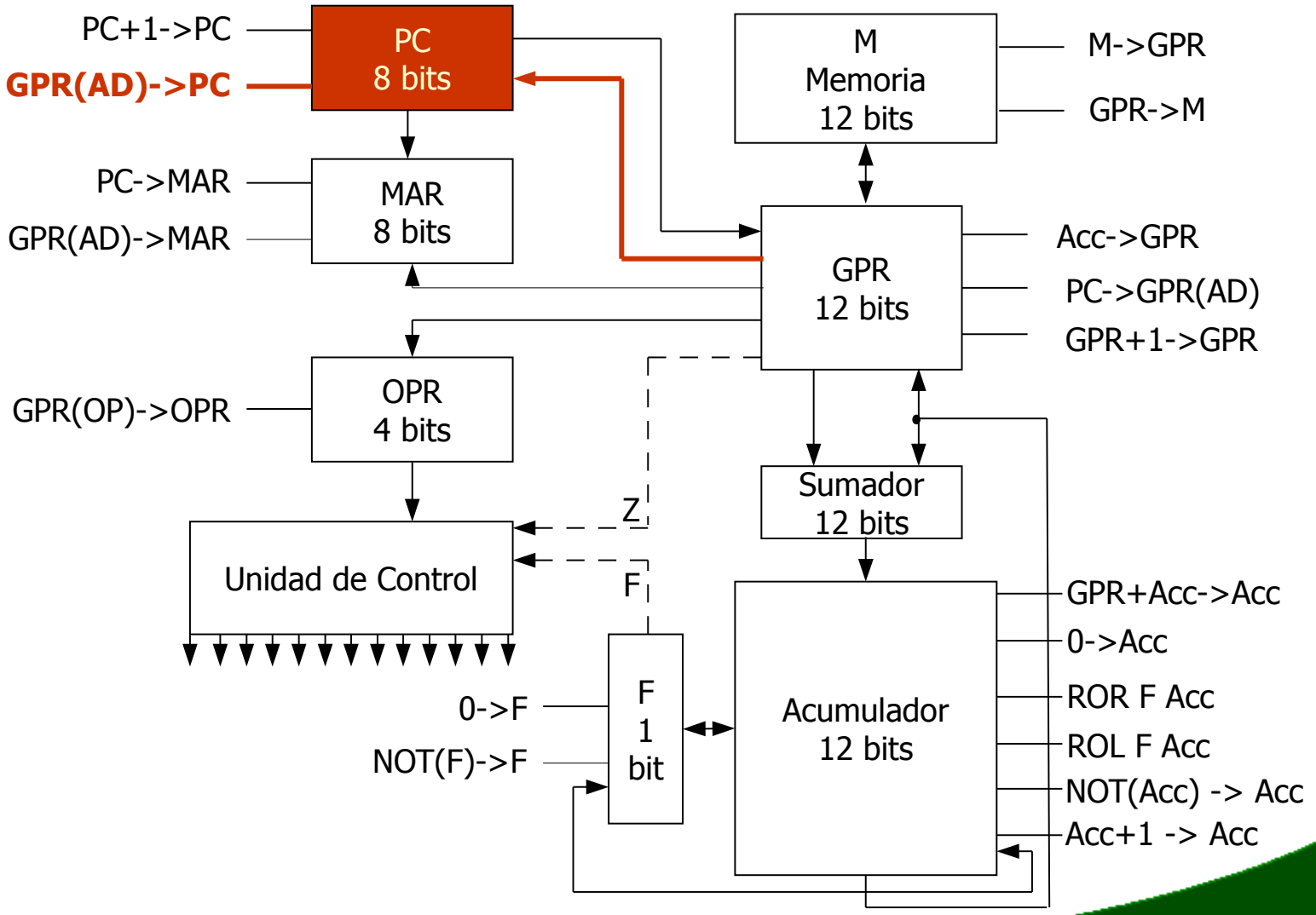
Computadora Mejorada: JMPI d



Computadora Mejorada: JMPI d



Computadora Mejorada: JMPI d

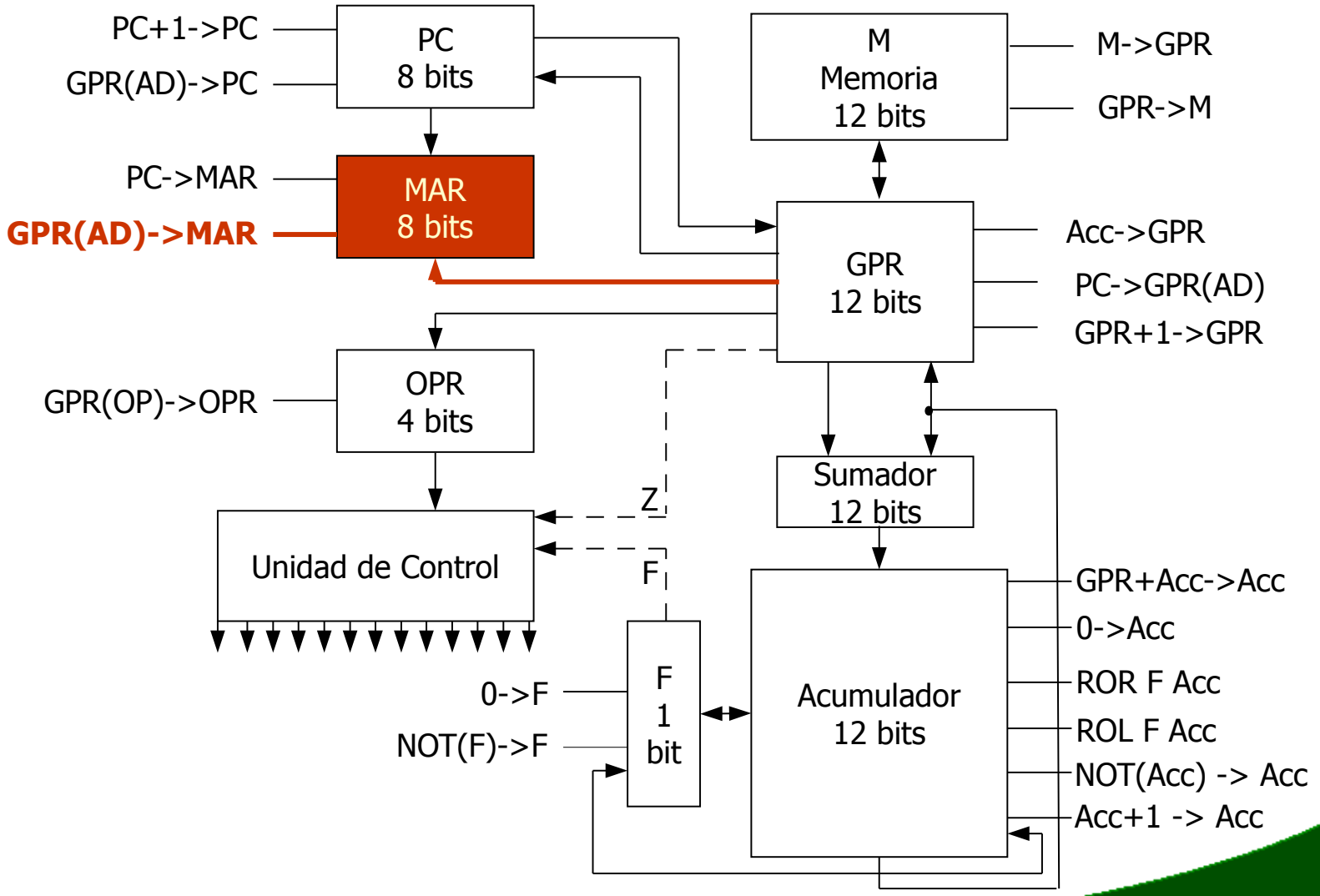


Computadora Mejorada: CSR d

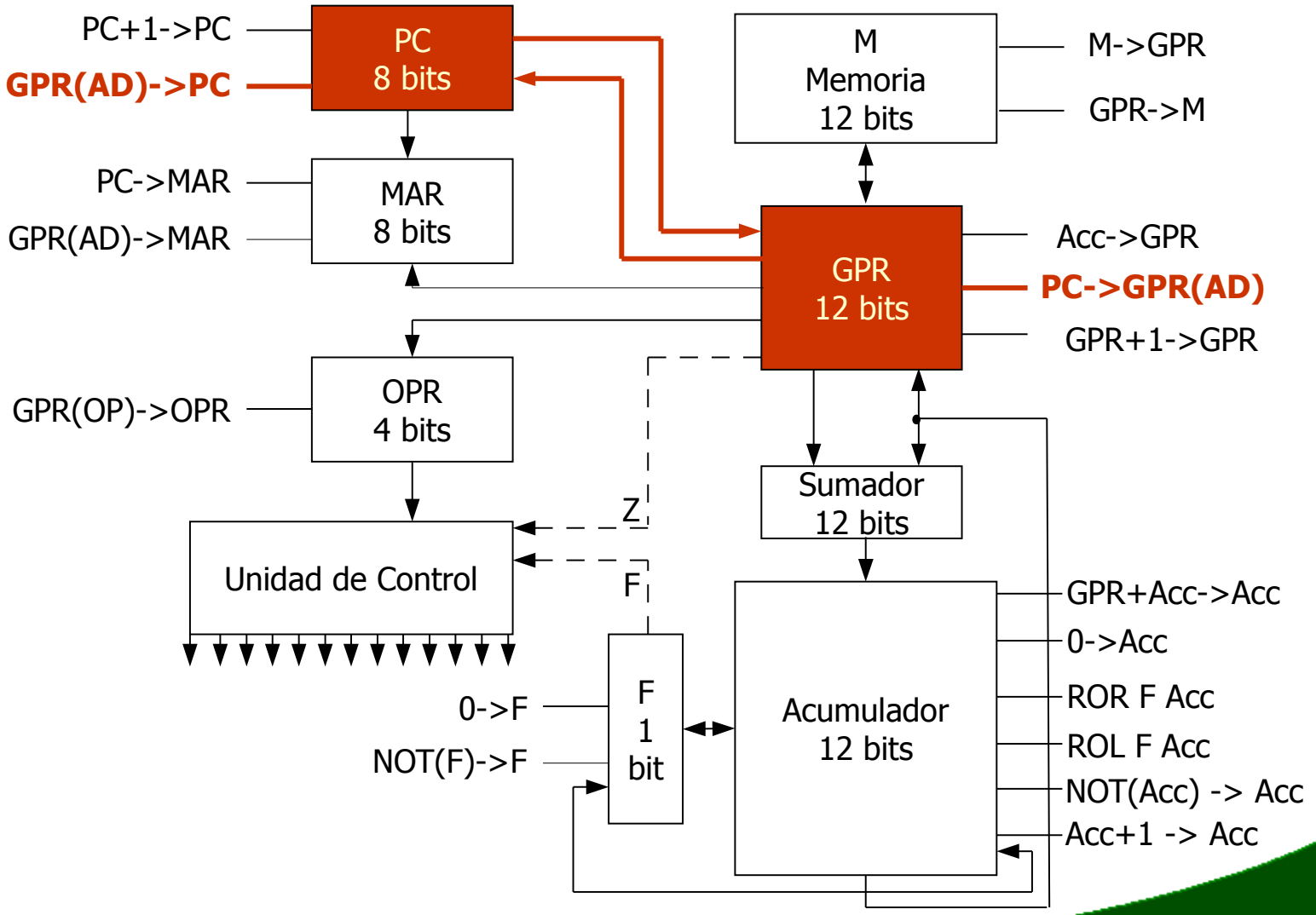
- CSR dirección
  - Llamada a subrutina
  - Almacena la dirección de retorno (PC+1) en la posición de memoria indicada en el campo de operando (d) y bifurca a la posición de memoria siguiente a ésta (d+1).
  - Secuencia de Microoperaciones:

Ciclo	Operación	Explicación
1	GPR(AD)->MAR	Transfiere la dirección de memoria donde está la dirección de memoria que almacena el valor desde el campo de operando (que se encuentra en los 8 bits inferiores del GPR) al MAR.
2	GPR(AD)->PC PC->GPR(AD)	Intercambia PC y los 8 bits menos signficativos de GPR. (Los 4 bits superiores de GPR pasan a valer 0).
3	GPR(AD)->M	Transfiere el GPR a memoria. En GPR está el PC incrementado, que será la dirección de retorno, y se almacena en la dirección de memoria indicada en el operando.
4	PC+1->PC	Se incrementa en 1 el PC para ir a la siguiente dirección de memoria de la indicada en el campo del operando. (Se podría hacer simultáneamente en el ciclo anterior).

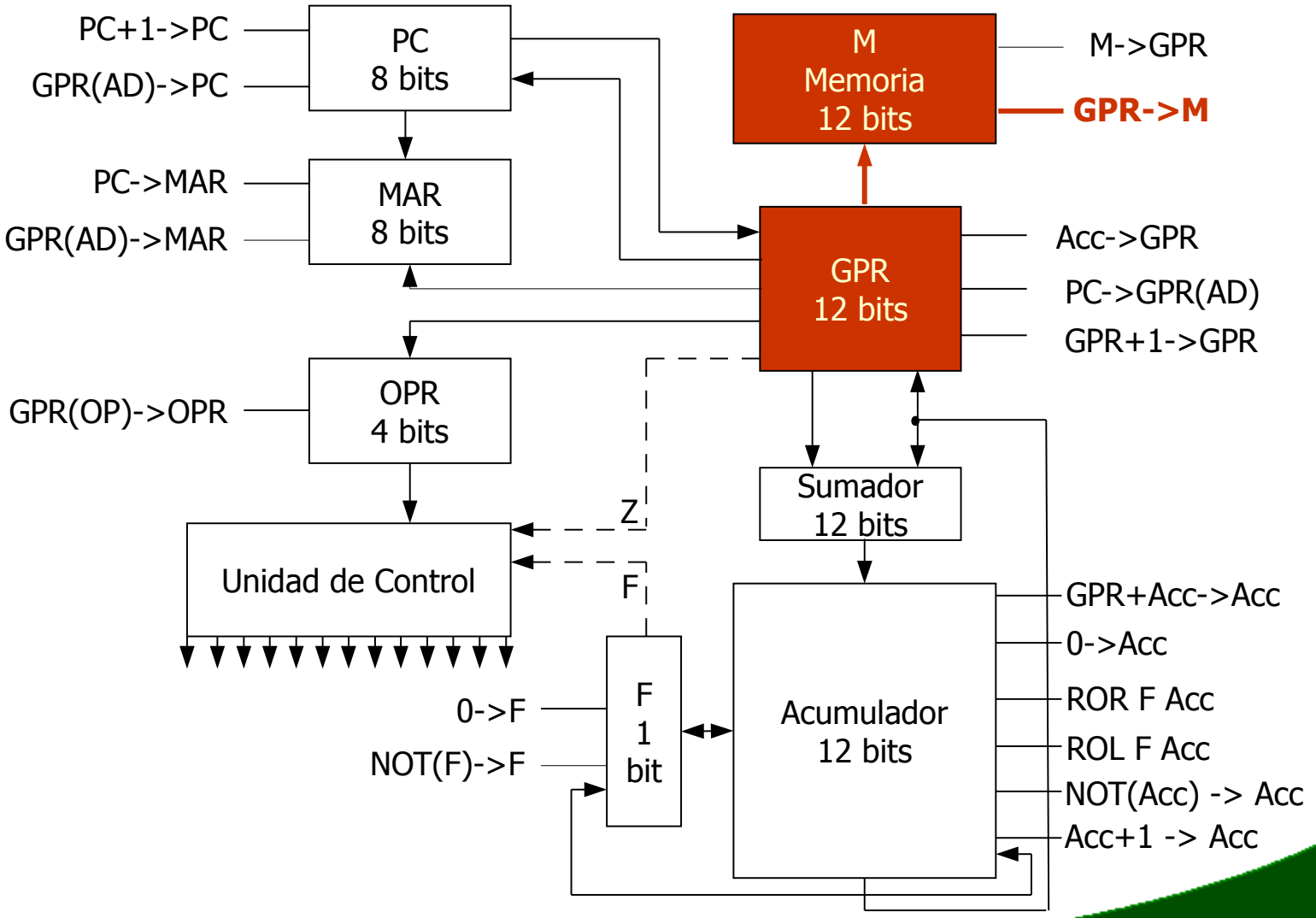
Computadora Mejorada: CSR d



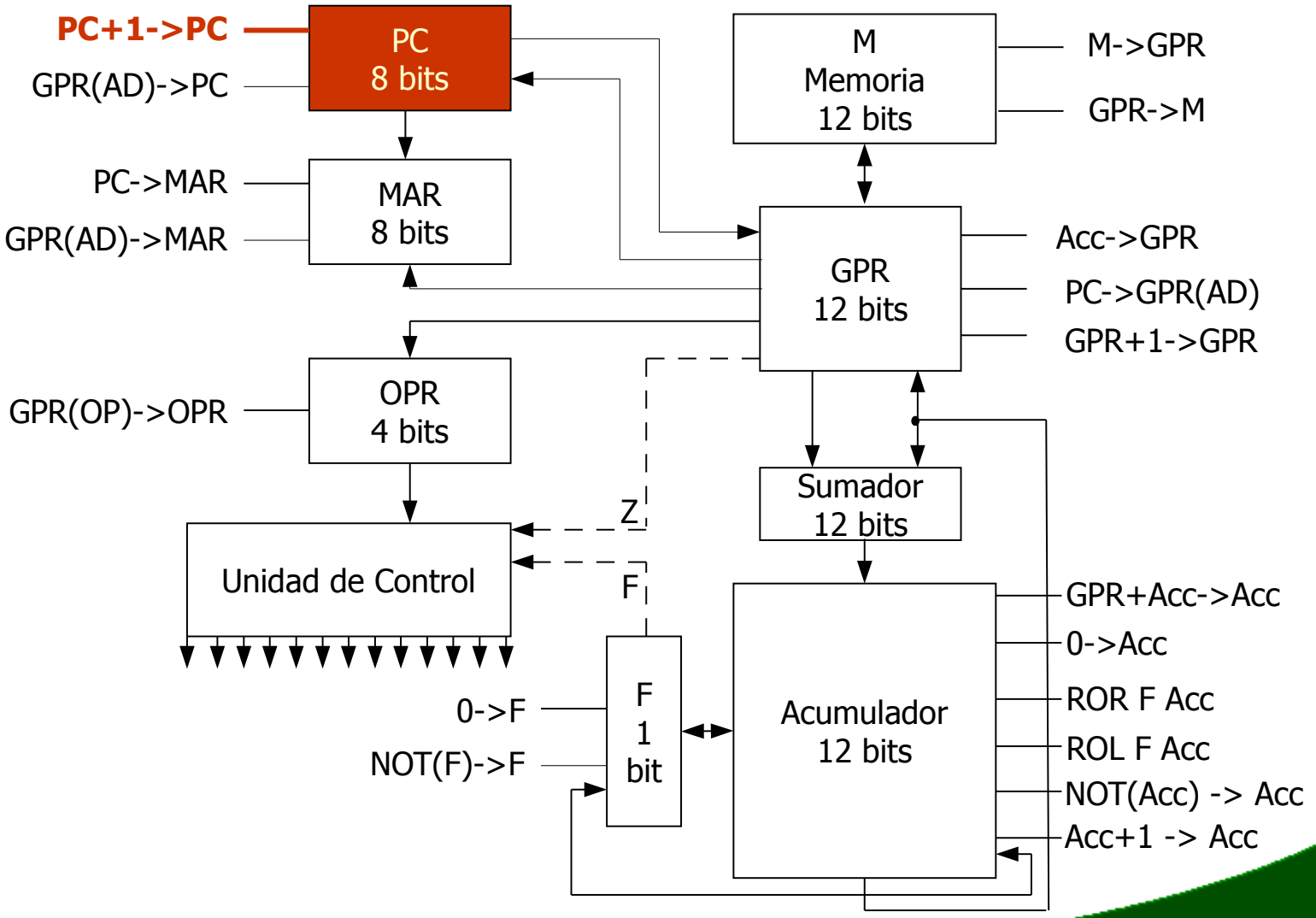
Computadora Mejorada: CSR d



Computadora Mejorada: CSR d



Computadora Mejorada: CSR d

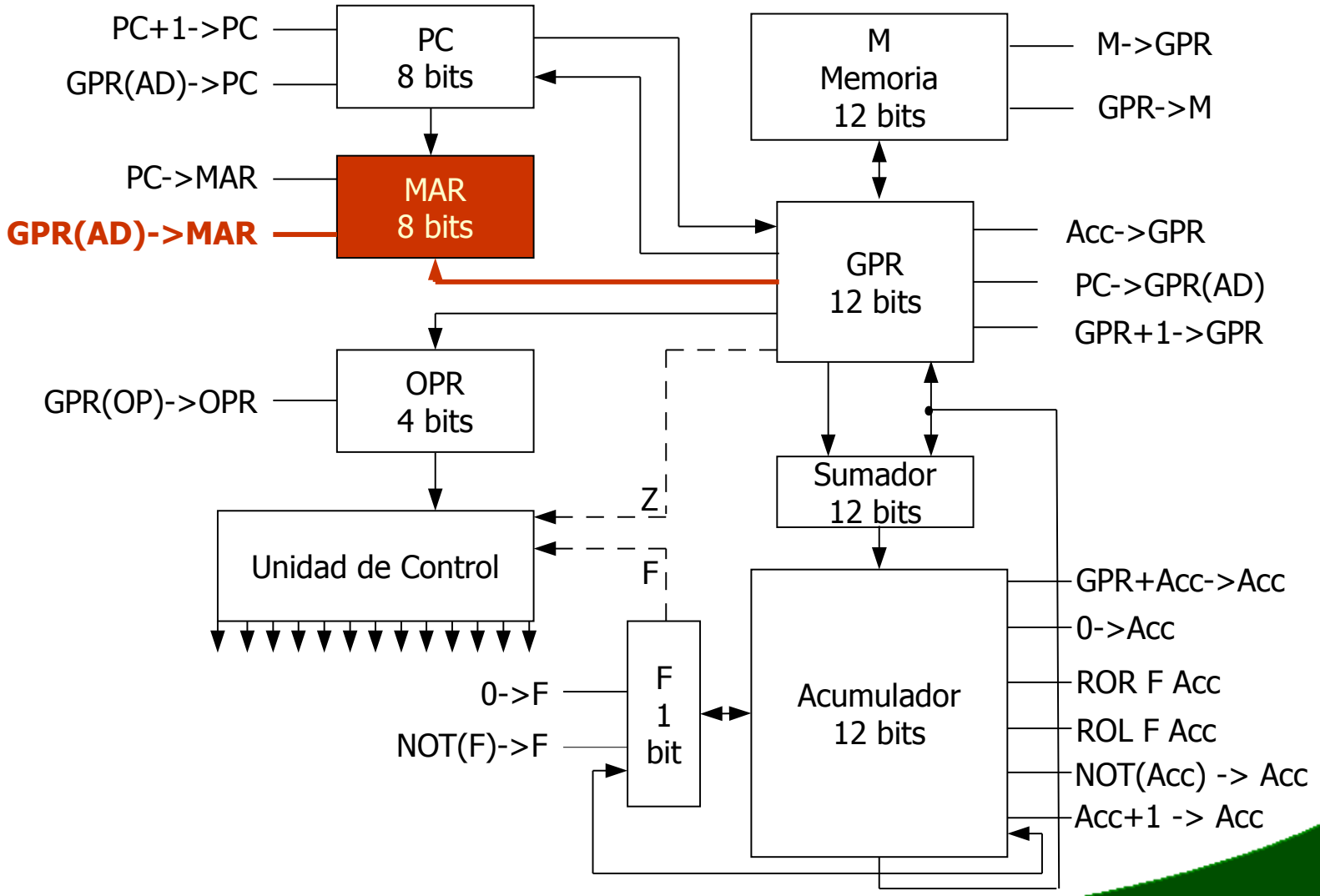


Computadora Mejorada: ISZ d

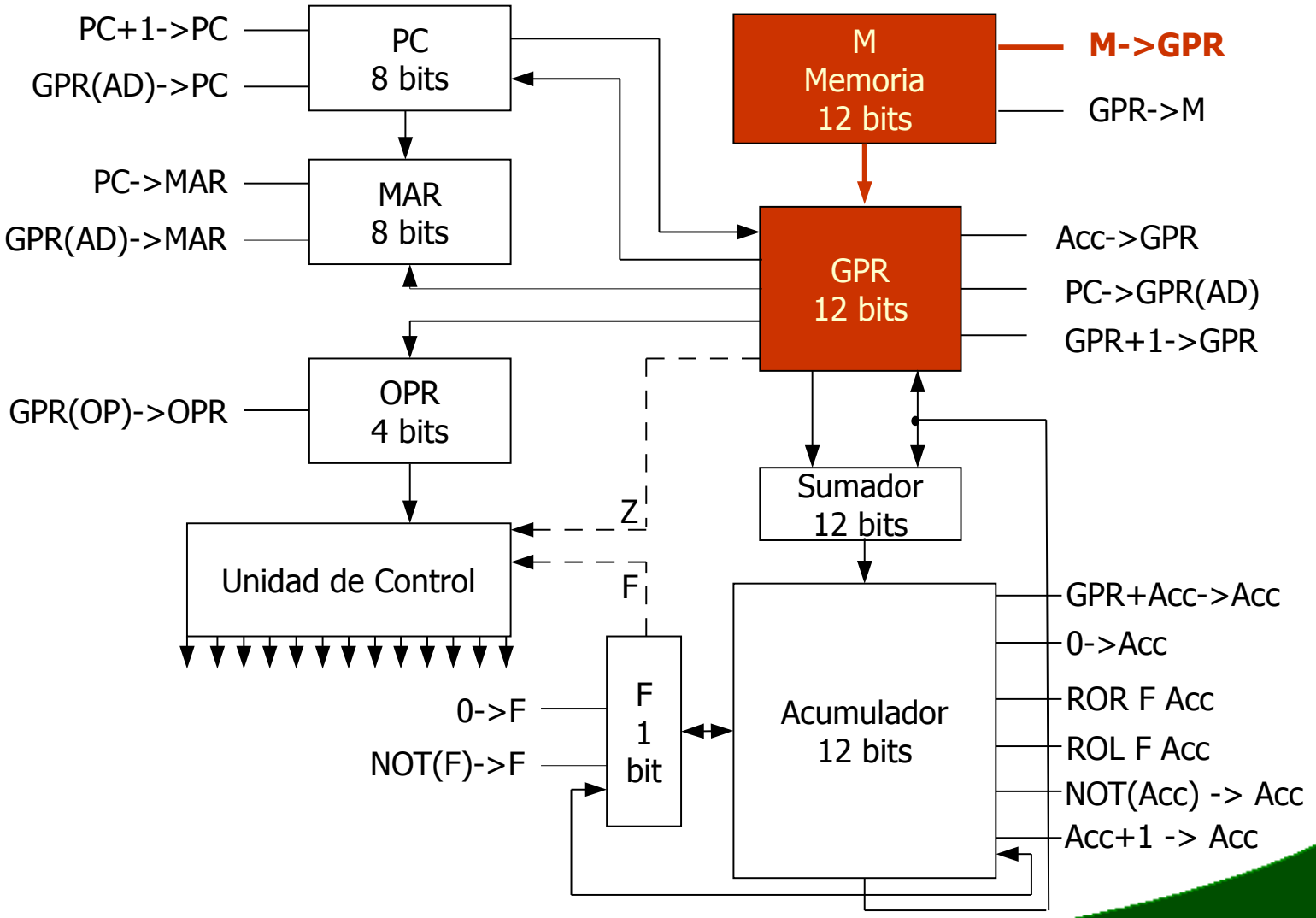
- ISZ dirección
  - Incrementa el contenido de una posición de memoria y si, al incrementar, dicho valor es 0, se salta la siguiente instrucción.
  - Secuencia de Microoperaciones:

Ciclo	Operación	Explicación
1	GPR(AD)->MAR	Transfiere la dirección de memoria donde está la dirección de memoria que almacena el valor desde el campo de operando (que se encuentra en los 8 bits inferiores del GPR) al MAR.
2	M->GPR	Lee desde la memoria la dirección que se encontraba en el campo de operando y la dirección de memoria se deposita en GPR
3	GPR+1->GPR	Incrementamos GPR
4	GPR->M	Almacenamos el valor GPR incrementado en la misma posición de memoria.
5	PC+1->PC (si Z=1)	Si GPR = 0, entonces Z=1, e incrementamos en ese caso PC; en caso contrario, no hacemos nada.

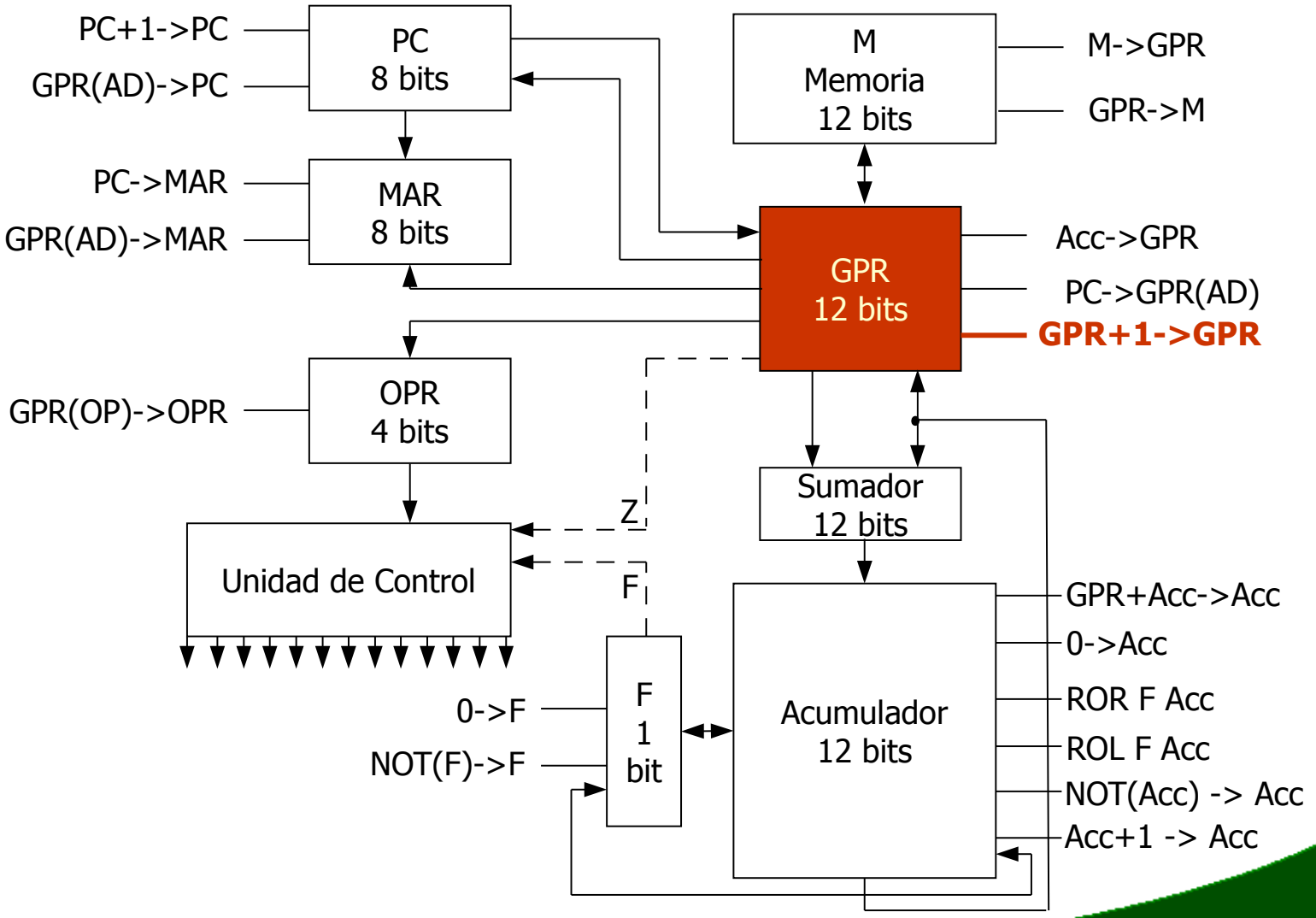
Computadora Mejorada: ISZ d



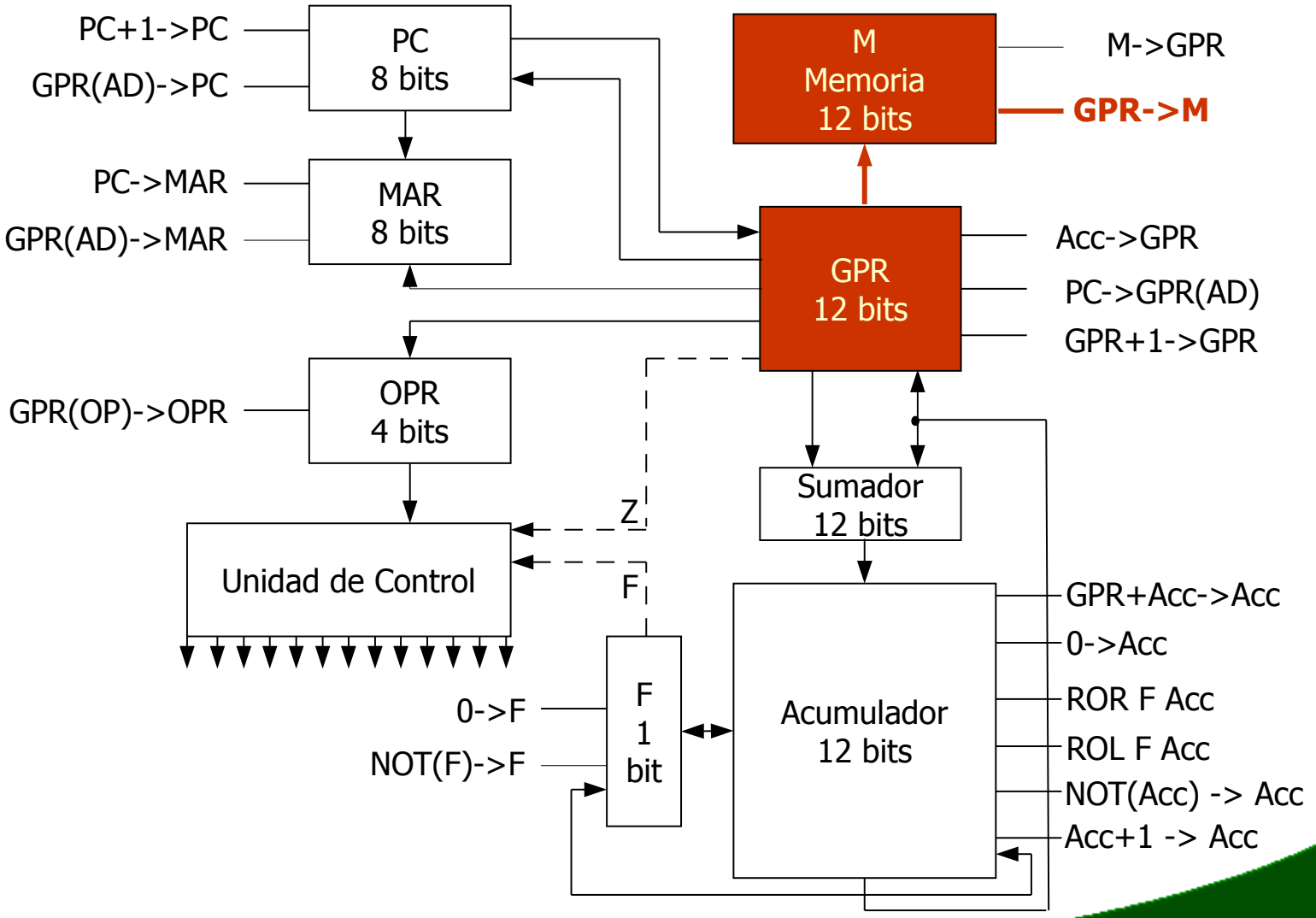
Computadora Mejorada: ISZ d



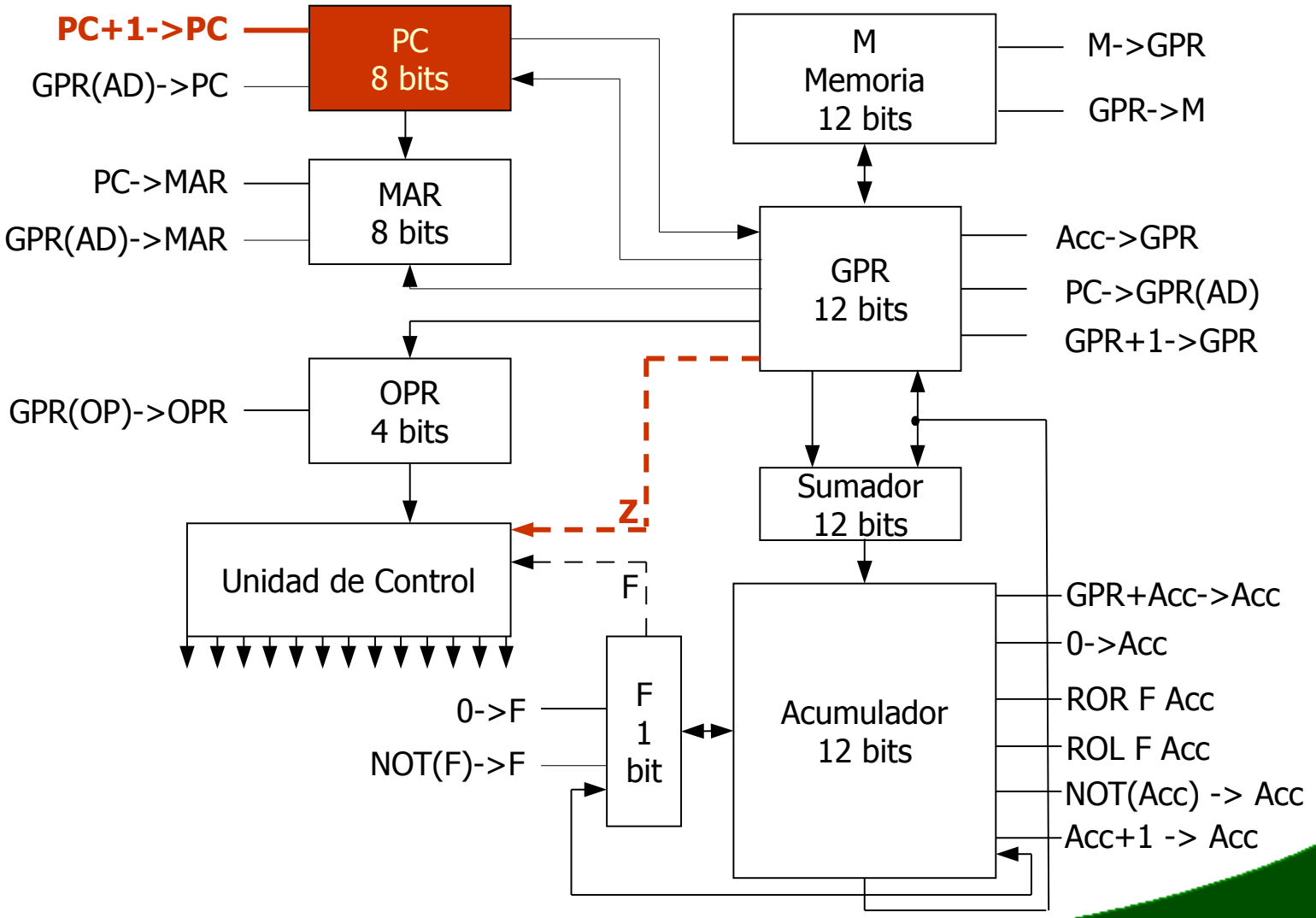
Computadora Mejorada: ISZ d



Computadora Mejorada: ISZ d



Computadora Mejorada: ISZ d



Computadora Mejorada: Programas

- Programa que implementa 2+3-5 y el resultado se almacenará en memoria en una variable "resultado".
  - Se definirán 4 variables: op1 = 2; op2 = 3; op3 = 5; resultado=0;

Dirección	Etiqueta	Instrucción Simbólica/	Instrucción	Explicación
0		CRA	CRA	Limpiamos el Acc (Acc = 0)
1		ADD op3	ADD A	Cargamos el valor 5 (variable op3, dirección de memoria A)
2		CTA	CTA	Complementamos (con estas 2 instrucciones hacemos el)
3		ITA	ITA	Incrementamos (complemento a 2, el negativo del 5)
4		ADD op1	ADD 8	Sumamos el valor 2 (variable op1, dirección de mem. 8)
5		ADD op2	ADD 9	Sumamos el valor 3 (variable op2, dirección de mem. 9)
6		STA resultado	STA B	Almacena el resultado en la variable "resultado" (dir. memoria B)
7		HLT	HLT	Para el programa
8	op1	0002		
9	op2	0003		
A	op3	0005		
B	resultado	0000		

Computadora Mejorada: Programas

- **Programa que calcula la suma de un vector de 3 valores, el resultado final en el Acumulador.**
  - Variables: vector = 1,2,3; indice = 3; puntero = &(vector)

Dirección	Etiqueta	Instrucción Simbólica/	Instrucción	Explicación
0		CRA	CRA	Limpiamos el Acc (Acc = 0)
1		ADD indice	ADD E	Cargamos el valor 3 (variable indice, posición Eh)
2		CTA	CTA	Complementamos (con estas 2 instrucciones hacemos el)
3		ITA	ITA	Incrementamos (complemento a 2, el negativo del 3)
4		STA indice	STA E	Guardamos el 3 en negativo en indice (posición Eh)
5		CRA	CRA	Limpiamos el Acc (Acc = 0)
6	bucle	ADDI puntero	ADDI F	Sumamos el valor al que apunta puntero (posición Fh)
7		ISZ puntero	ISZ F	Incremento el puntero, apunto al siguiente valor del vector.
8		ISZ indice	ISZ E	Incremento el indice; si es cero, me salto la siguiente instrucción
9		JMP bucle	JMP 6	Saltar a procesar el siguiente elemento (posición 6h)
A	fin	HLT	HLT	Para el programa
B	vector	1		
C		2		
D		3		
E	indice	3		
F	puntero	B		

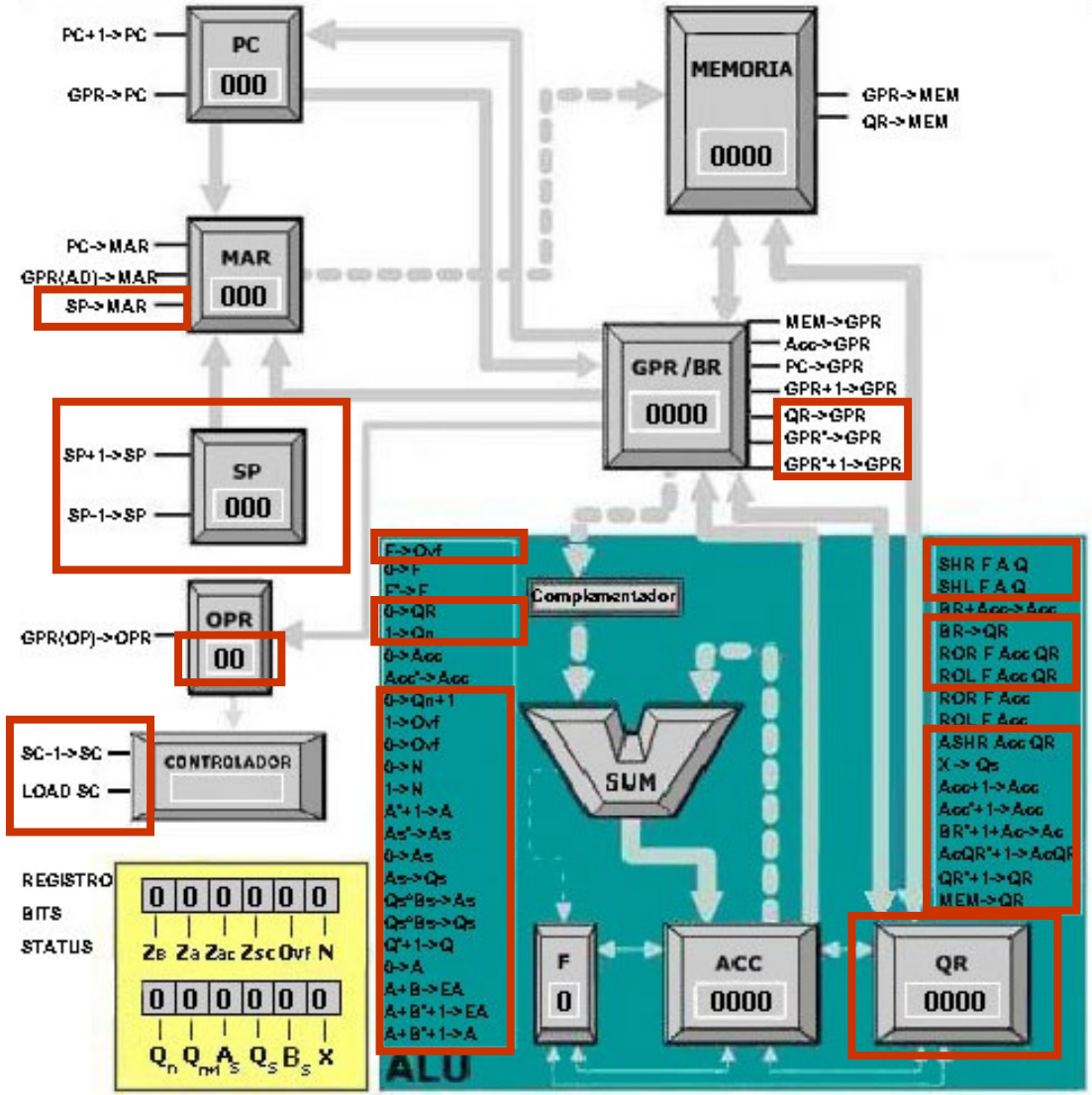
Ampliaciones en la Computadora Mejorada

- Limitaciones:
  - Número muy limitado de microoperaciones.
  - Número limitado de instrucciones.
  - No tiene Pila.
  - Registros adicionales y con mayor funcionalidad.
  - Registros de mayor tamaño.
  - Más bits de estatus.
  - Unidades de control con contadores de bucle.

SiCoMe 2.1

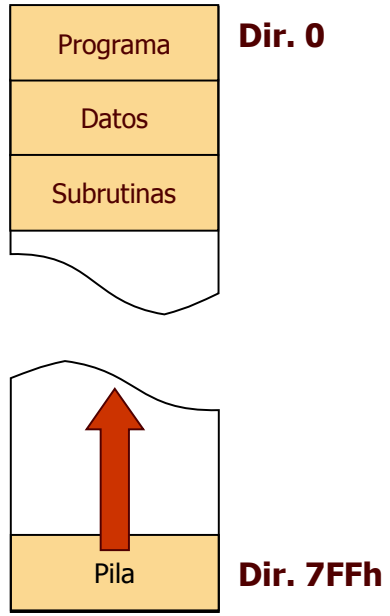
- Simulador de la Computadora Mejorada con ampliaciones.
  - Se ha aumentado el número de instrucciones posibles (16->32)
  - Se ha aumentado la memoria a 11 bits de dirección (256->2048)
  - Tiene un registro Puntero de Pila (*SP, Stack Pointer*)
  - Se ha incluido un registro (QR) extra, capaz de acceder directamente a memoria.
  - Se han incluido muchas más microoperaciones (18->56)
  - Se han aumentado el conjunto de bits de estado (2->12)
  - Se han incluido microoperaciones que permiten realizar operaciones con datos en complemento a 2 o en signo-magnitud
  - Se ha incluido un Registro Contador dentro de la Unidad de Control para implementar bucles con repeticiones determinadas.
  - Se puede definir completamente la tabla de control de bifurcación.

SiCoMe 2.1


PC+1->PC
GPR->PC
PC
000
MEMORIA
0000
GPR->MEM
QR->MEM
PC->MAR
GPR(AD)->MAR
SP->MAR
MAR
000
MEM->GPR
Acc->GPR
PC->GPR
GPR+1->GPR
GPR/BR
0000
QR->GPR
GPR*->GPR
GPR*+1->GPR
SP+1->SP
SP-1->SP
SP
000
F->Ovf
0->F
F*->F
0->QR
1->Qn
0->Acc
Acc*->Acc
Complementador
SHR F A Q
SHL F A Q
BR->QR
ROR F Acc QR
ROL F Acc QR
ROR F Acc
ROL F Acc
GPR(OP)->OPR
OPR
00
0->Qn+1
1->Ovf
0->Ovf
0->N
1->N
A*+1->A
As*->As
0->As
As->Qs
Qs*Bs->As
Qs*Bs->Qs
Q*+1->Q
0->A
A+B->EA
A+B*+1->EA
A+B*+1->A
SUM
ASHR Acc QR
X -> Qs
Acc+1->Acc
Acc*+1->Acc
BR*+1+Ac->Ac
AcQR*+1->AcQR
QR*+1->QR
MEM->QR
SC-1->SC
LOAD SC
CONTROLADOR
REGISTRO
BITS
STATUS
0 0 0 0 0 0
Zb Za Zac Zsc Ovf N
0 0 0 0 0 0
Qn Qn+1 As Qs Bs X
F
0
ACC
0000
QR
0000
ALU

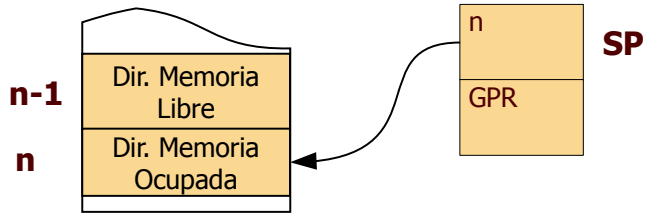
Pila

- Pila
  - Implementación sistema LIFO (*Last In, First Out*):
    - Utilización de un registro que almacena la dirección de memoria donde se colocó el último valor almacenado. (Apunta a una dirección ocupada).
    - La memoria pila se utiliza en pasos descendentes (desde las posiciones más altas de la memoria hacia las más bajas de la memoria) => Evitar colisionar con el espacio de programa, que suele estar en las direcciones bajas de la memoria.
  - Registro SP:
    - SP se utiliza como registro similar al PC, para almacenar la dirección de memoria del último dato introducido en la memoria pila.
    - Para acceder a la memoria pila hay que pasar el contenido de SP al MAR: SP->MAR
    - SP puede ser incrementado o decrementado.

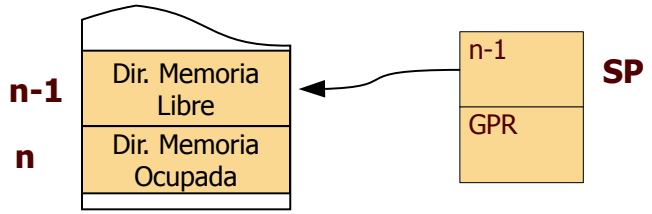


PUSH

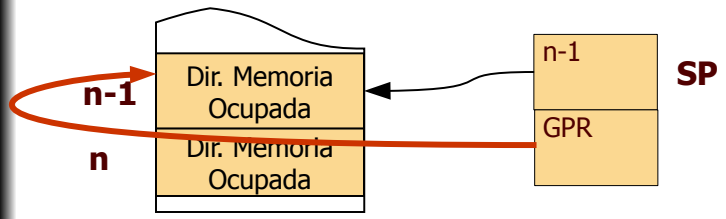
- Introducir un elemento en la pila (*Push*)



- Inicialmente SP "apunta" a la última posición de memoria ocupada.



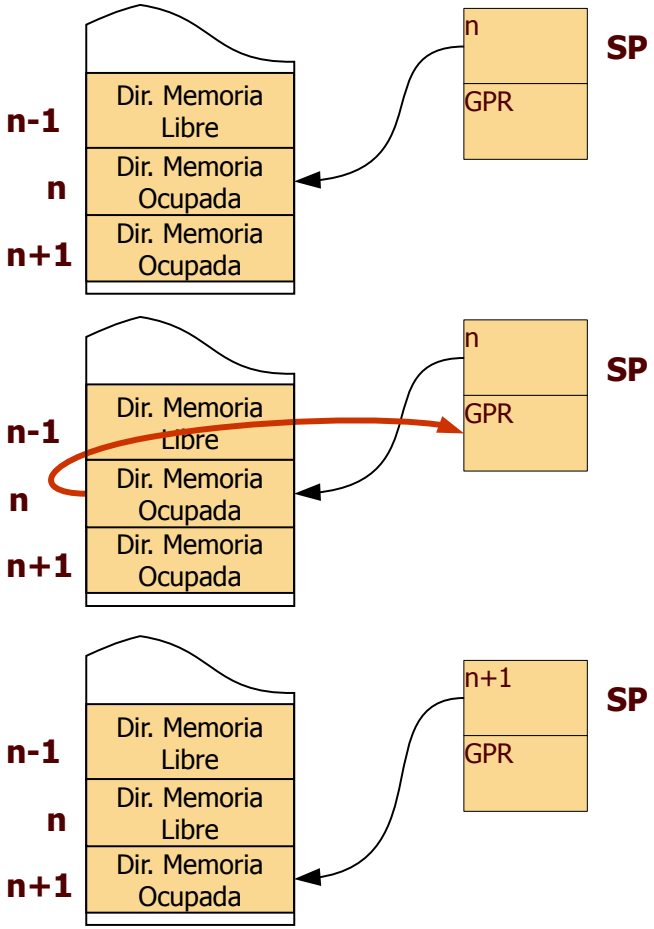
- Decrementamos la posición ("apuntando" a una dirección "libre").



- Almacenamos en la nueva posición "libre" el elemento de GPR, pasando SP a apuntar ahora a una posición "ocupada".

POP

- Sacar el último elemento de la pila (*Pop*)

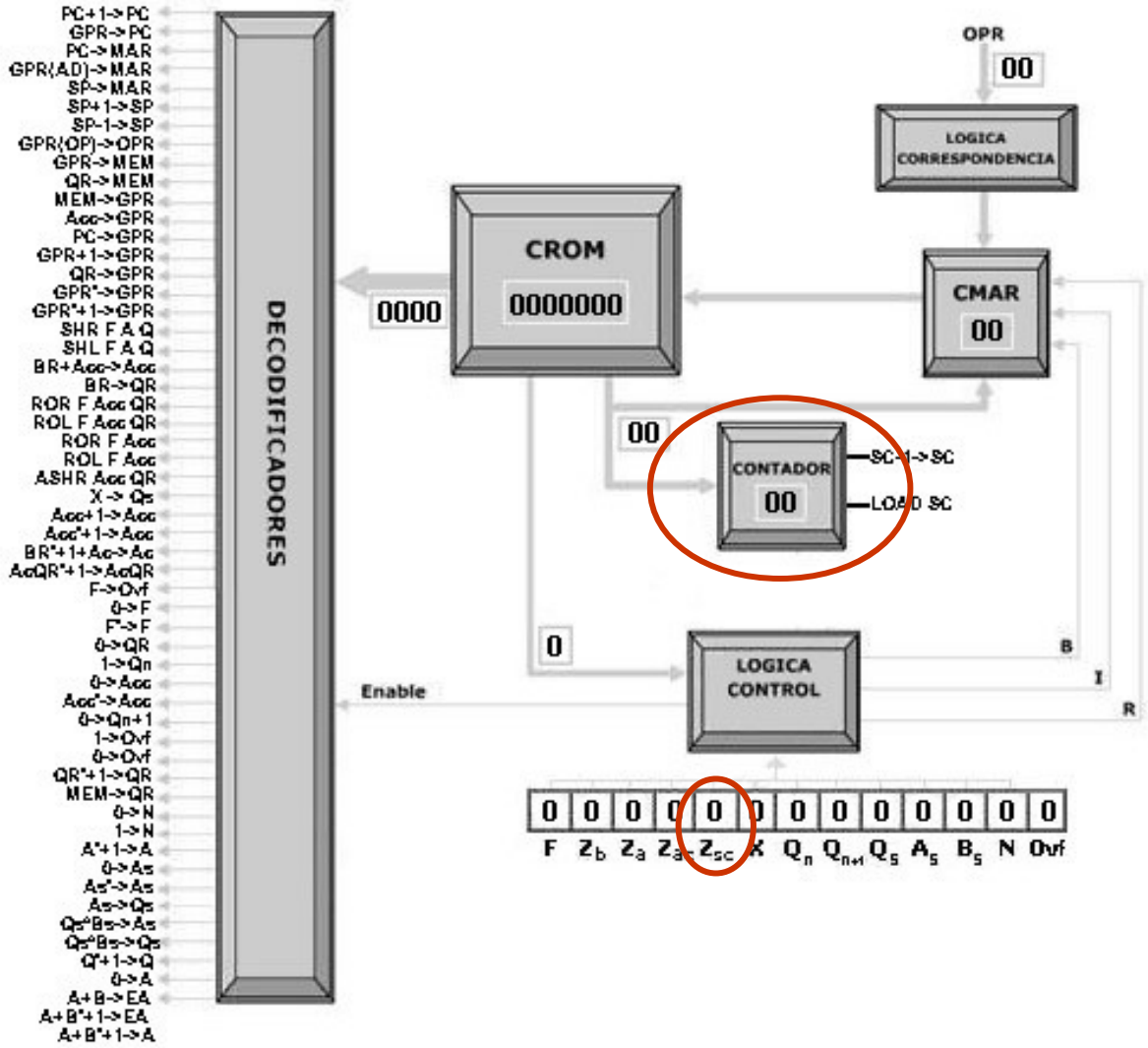


- Inicialmente SP "apunta" a la última posición de memoria ocupada.
- Leemos de la posición de memoria a la que apunta SP, guardamos el valor en GPR. Dicha posición, pasa a estar "libre"
- Incrementamos la posición del SP ("apuntando" a una dirección "ocupada").

Unidad de Control con Contador de Bucles

- Para la ejecución repetitiva de microoperaciones, se introduce un contador SC (*Sequence Counter*).
  - Microoperaciones:
    - "SC-1->SC": Decrementar el contador.
    - "Load SC": Carga paralelo.
  - Salidas:
    - "Z_{SC}" : Es 1 cuando el contador alcanza 0. En otro caso, vale 0.
- Microprogramación:
  - El valor de carga paralelo se introduce solapado en el campo de dirección de bifurcación.
- Cableado:
  - El valor de carga paralelo se introduce cableando el valor bit a bit.

Unidad de Control con Contador de Bucles



Unidad de Control con Contador de Buces