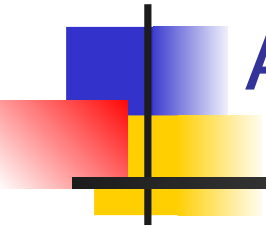


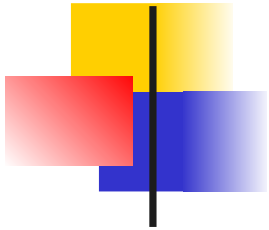
Arquitectura de Computadoras
ALU (Arithmetic Logic Unit)

CARACTERISTICAS FUNDAMENTALES I

- Es la parte de la computadora que realiza las operaciones aritméticas y lógicas
- Los otros elementos del sistema están principalmente para traer datos a la ALU o tomar sus resultados
- Se utilizan registros como fuentes y destinos para la mayoría de las operaciones de la ALU
- En las primeras máquinas, la simplicidad y confiabilidad determinaban la estructura de la CPU y su ALU. Por ello:
  - Las máquinas fueron construidas alrededor de un único registro (el ACUMULADOR)
  - El ACUMULADOR se utilizaba en casi todas las instrucciones de la ALU
- La potencia y flexibilidad de la CPU y la ALU mejoran con la complejidad del hardware
  - Se usan conjuntos de registros de uso general para almacenar operandos, direcciones y resultados
  - Se incrementan las capacidades de la ALU
  - Se usa hardware especial para soportar la transferencia de ejecución entre puntos en un programa
  - Se replican unidades funcionales dentro de la ALU para permitir la ejecución concurrente

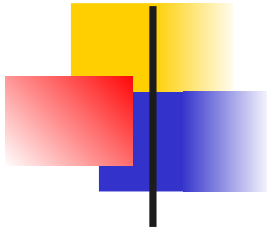
CARACTERISTICAS FUNDAMENTALES II

- El problema consiste en diseñar una ALU completamente funcional al mínimo costo
- Las decisiones de diseño incluyen:
  - Tipos de datos
  - Repertorio de operaciones (cuántos operadores, cuáles (que harán), cuán complejos)
  - Formatos de la instrucción (longitud del campo de operando, cantidad de direcciones)
  - Registros (número de registros y operaciones realizables sobre los mismos)
  - Modos de direccionamiento
  - RISC vs CISC



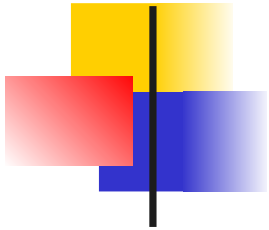
TIPOS DE DATOS

- Direcciones
- Números
  - Enteros (distintas magnitudes)
  - Punto Flotante (distintas magnitudes y precisiones)
  - Decimales (empaquetados y no empaquetados)
- Cadenas de caracteres
- Lógicos (bits)



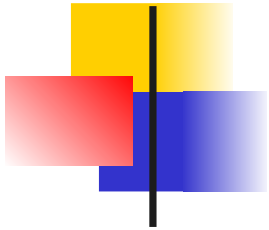
EL REPERTORIO DE INSTRUCCIONES

- Gran parte de la organización / arquitectura de la computadora es invisible al programador de alto nivel
  - En cierto sentido el programador no debe preocuparse por cómo es realmente la arquitectura subyacente
- El conjunto de instrucciones o repertorio de instrucciones es la frontera donde el diseñador de la computadora y el programador pueden ver la misma máquina
- Por eso, un examen del repertorio de instrucciones permite explicar la CPU
- Lo que se investiga en esta sección es como se diseña el conjunto de instrucciones y el impacto que tiene sobre el sistema completo



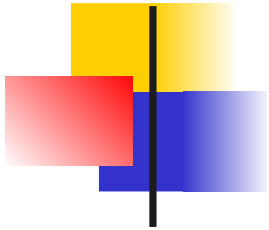
CONTENIDO DE LA INSTRUCCION

- Cada instrucción debe contener 4 elementos básicos:
  - El código de operación, que especifica la operación que debe ser realizada
  - Referencias a los operandos fuentes (donde están los datos requeridos para ejecutar la operación)
  - Referencia al resultado (donde debe ser colocado el resultado de la operación)
  - Referencia a la próxima instrucción: donde se encuentra la próxima instrucción que debe ejecutarse
- En la mayoría de los casos se prescinde del último elemento a costa de:
  - Asumir que las instrucciones se ubican en secuencia física
  - Agregar a la CPU un **Registro de Próxima Instrucción (RPI) o Program Counter** que se actualiza automáticamente durante el ciclo de búsqueda de la instrucción
  - Agregar al repertorio de instrucciones un conjunto de instrucciones de bifurcación condicional e incondicional
- También puede prescindirse de la referencia al resultado asumiendo que el mismo se conserva en uno de los dos operandos fuentes
- Puede prescindirse de la referencia a uno de los operandos fuentes si se cuenta con un Acumulador (operando implícito)
- Finalmente puede prescindirse totalmente de operandos (pila o registros implícitos)



REPERTORIO DE INSTRUCCIONES

- El repertorio de instrucciones debe ser funcionalmente completo, permitiendo al usuario formular cualquier tarea de procesamiento de alto nivel.
- Una posible clasificación contempla cinco categorías de instrucciones:
  - Operaciones aritméticas
  - Operaciones lógicas
  - Movimientos de datos (dentro del sistema central)
  - Operaciones de entrada / salida (movimientos de datos entre el sistema central y los dispositivos periféricos)
  - Operaciones de control
- Se han diseñado repertorios de instrucciones con
  - Un número pequeño de instrucciones
  - Cientos de instrucciones
- La tendencia actual es usar el número de instrucciones suficiente para hacer bien el trabajo (RISC vs CISC)



REPERTORIO DE INSTRUCCIONES (II)

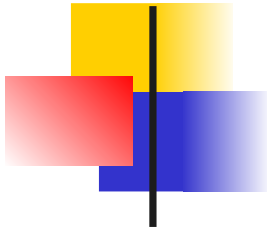
- Hasta los ochenta (1980) la tendencia fue construir repertorios de instrucciones más y más complejos, conteniendo cientos de instrucciones y variaciones
- Lo que se intentaba era proveer mecanismos que sirvieran de puente a la brecha semántica entre el funcionamiento a bajo y alto nivel de una computadora
  - Reconciliando las visiones del programador de lenguajes de alto nivel y el programador a nivel de ensamblador
  - Suministrando un repertorio de instrucciones rico, en un intento de acercarse al estilo del programador de lenguajes de alto nivel
  - Permitiendo al compilador puentear la brecha con una instrucción simple más que recurriendo a un conjunto de instrucciones
  - Lamentablemente, no siempre tuvieron el efecto deseado

REPERTORIO DE INSTRUCCIONES (III)

- Se dice que el escritor de compiladores es el mejor arquitecto de diseño, debido a que ha tenido que tratar con pobres decisiones de arquitectura
- Los atributos de un buen repertorio de instrucciones son:
  - Completo: permitiendo la construcción de un programa a nivel de máquina convencional para evaluar cualquier función computable
  - Eficiente: las funciones realizadas más frecuentemente deben ser resueltas rápidamente con pocas instrucciones
  - Clases de instrucciones regulares y completas: suministrando conjuntos “lógicos” de operaciones
  - Ortogonal: definiendo las instrucciones, los tipos de datos y los modos de direccionamiento independientemente
- Adicionalmente, debe ser compatible con el hardware y el software existente en una línea de productos

DIRECCIONES EN LA INSTRUCCION

- Cada operando en una instrucción significa una dirección
- En una operación aritmética o lógica típica se necesitan tres direcciones (dos datos fuente y un resultado)
- Estas direcciones pueden figurar explicitamente en la instrucción o estar implícitas
- Instrucciones de tres direcciones
  - Ambos operandos fuente así como el resultado están explícitamente indicados en la instrucción
  - Ejemplo: X = Y + Z
  - Con la velocidad de la memoria acercándose a la del procesador (debido a el *caching*), esta variante le da una gran flexibilidad al compilador evitando la molestia de guardar resultados en el conjunto de registros internos de la CPU y usando la memoria con un gran conjunto de registros
  - De todos modos el método es raramente utilizado porque produce instrucciones muy largas



DIRECCIONES EN LA INSTRUCCION (II)

- Instrucciones de dos direcciones
  - Una de las direcciones se utiliza para especificar tanto uno de los operandos fuente como el resultado
  - Ejemplo: X = X + Y
  - Es un formato habitual en los repertorios de instrucciones
- Instrucciones de una dirección
  - Dos direcciones están implícitas en la instrucción (la de uno de los operandos fuente y la del resultado)
  - Son las tradicionales operaciones basadas en un Acumulador
  - Ejemplo: Acc = Acc + X
- Instrucciones sin direcciones
  - Todas las direcciones están implícitas
    - Operaciones basadas en registros
      - Ejemplo TBA (transfiere el contenido del registro B al registro A)
    - Operaciones basadas en una pila
      - Las operaciones utilizan una pila en memoria para almacenar los operandos
      - Se interactúa con la pila con operaciones PUSH y POP

DIRECCIONES EN LA INSTRUCCION (III)

VENTAJAS Y DESVENTAJAS

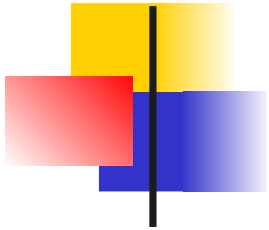
- Un menor número de direcciones en una instrucción resulta en:
  - Instrucciones más primitivas
  - Una CPU menos compleja
  - Instrucciones más cortas
  - Mayor número de instrucciones en un programa
  - Programas más largos y complejos
  - Mayores tiempos de ejecución

DIRECCIONES EN LA INSTRUCCION (IV)

Considere la necesidad de resolver Y = (A-B) / (C+D*E)

Con tres direcciones	Con dos direcciones	Con una dirección	Sin direcciones (Y = AB-CDE*+/)

SUB Y,A,B
MUL T,D,E
ADD T,T,C
DIV Y,Y,T
	MOV Y,A
	SUB Y,B
	MOV T,D
	MUL T,E
	ADD T,C
	DIV Y,T
		LOAD D
		MUL E
		ADD C
		STORE Y
		LOAD A
		SUB B
		DIV Y
		STORE Y
			PUSH A
			PUSH B
			SUB
			PUSH C
			PUSH D
			PUSH E
			MUL
			ADD
			DIV
			POP Y

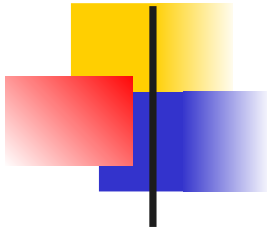


OPERACIONES DE CONTROL

- Branch
  - Condicional o incondicional
    - Modifica el PC y provoca un salto a una parte del programa distinta de la instrucción que sigue en la secuencia física
  - Skip
    - La dirección de bifurcación está implícita (normalmente salta una instrucción).

    ISZ R1 /* Incremente y salte si es cero

    BRA TOP /* Bifurca incondicionalmente a TOP
  - Subroutine call / return
    - Salta a una subrutina con la expectativa de retornar y reasumir la operación en la próxima instrucción
    - Debe preservar la dirección de la próxima instrucción (dirección de retorno). Puede almacenarla en:
      - Un registro o una ubicación de memoria
      - Como parte de la subrutina (habitualmente al comienzo)
      - En una pila
    - Los parámetros pueden ser pasados a y desde una subrutina en forma similar
    - El uso de una pila es el único enfoque reentrante (permite la recursividad)
    - A cada rutina llamada le es asignado una porción (frame) de la pila que contiene las variables que serán pasadas, la dirección de retorno y los resultados que serán retornados



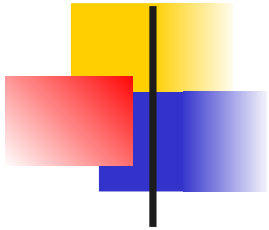
ENDIAN WARS

- Los arquitectos deben especificar como son almacenados los datos (el orden de los bytes) en la memoria y los registros
- El problema se conoce como “endian wars” y admite los dos enfoques siguientes:
  - Big endian
  - Little endian
- Considere el valor hexadecimal $12345678 y como se almacena en memoria comenzando en la dirección hexadecimal $100

Big endian (los bytes más significativos en las direcciones más bajas)	Little endian (en orden inverso)
100 12 101 34 102 56 103 78	100 78 101 56 102 34 103 12

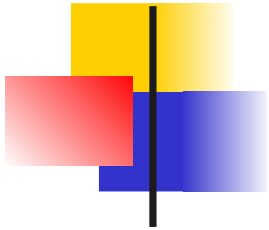
ENDIAN WARS

- Observaciones
  - Al almacenar varios elementos de dato en un segmento de memoria cada elemento debe tener la misma dirección (big o little endian no cambian esto)
  - El método elegido no tiene efecto en el ordenamiento de los elementos de una estructura de datos
  - No hay consenso general acerca de cual es el mejor sistema
    - Little endian se utiliza en Intel X86, Pentium, VAX
    - Big endian en S370, Motorola 680x0, RISCs
  - No hay ventajas reales en un estilo u otro
    - Normalmente la decisión está basada en la necesidad de soportar las máquinas previas
  - Los mayores problemas son
    - la transferencia de datos entre máquinas con distinto estilo que implica un proceso de conversión de formato
    - Manipulación de los bytes (o bits) individuales de una palabra de varios bytes



MODOS DE DIRECCIONAMIENTO

- Una vez que se ha determinado el número de direcciones contenido en una instrucción, debe determinarse la manera en la que cada campo de direcciones especifica la ubicación de memoria
- Lo que se necesita es la capacidad para referirse a un amplio rango de direcciones
- Hay compromisos entre
  - Rango de direcciones y flexibilidad
  - Complejidad del cálculo de la dirección efectiva



DIRECCIONAMIENTO INMEDIATO

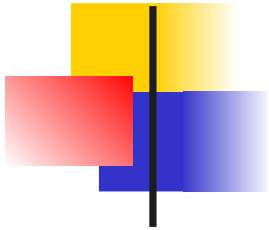
- **Modo Inmediato**
  - El operando está contenido en la instrucción
  - El dato es una constante al momento de la ejecución
  - No se requieren referencias adicionales a la memoria (además de las necesarias para obtener la instrucción)
  - El tamaño del operando (y en consecuencia el rango de valores) es limitado

DIRECCIONAMIENTO DIRECTO

- **Modo Directo**
  - El campo de direcciones de la instrucción contiene la dirección efectiva del operando
  - No se requieren cálculos
  - Se necesita un acceso adicional a la memoria para recuperar el dato
  - El rango de direcciones está limitado por el tamaño del campo que contiene la dirección
  - La dirección es una constante al tiempo de ejecución (no así el dato que puede ser modificado durante la ejecución del programa)
  - Algunas máquinas utilizan variantes del direccionamiento directo: direccionamiento directo y extendido sobre la 68HC11 – direcciones de 8 y 16 bits

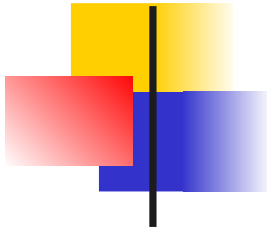
DIRECCIONAMIENTO INDIRECTO

- **Modo Indirecto**
  - El campo de direcciones en la instrucción especifica una dirección de memoria que contiene la dirección del dato
  - Se requieren dos accesos a memoria para recuperar el dato: el primero para buscar la dirección efectiva y el segundo para recuperar el dato
  - El rango de direcciones efectivas es 2^n donde n es la longitud de la palabra de dato
  - El número de direcciones que puede ser utilizado para mantener la dirección efectiva está limitado a 2^k, donde k es la longitud del campo de dirección en la instrucción



DIRECCIONAMIENTO EN BASE A REGISTRO

- **Modos de direccionamiento basados en registros**
  - Direccionamiento por registro: Es similar al directo, pero el campo de dirección especifica la dirección de un registro (número de registro)
  - Direccionamiento indirecto por registro: Similar al indirecto, pero el campo de dirección especifica el número de registro que contiene la dirección efectiva del dato
  - Se obtiene acceso más rápido a los datos y en la instrucción campos de dirección más pequeños



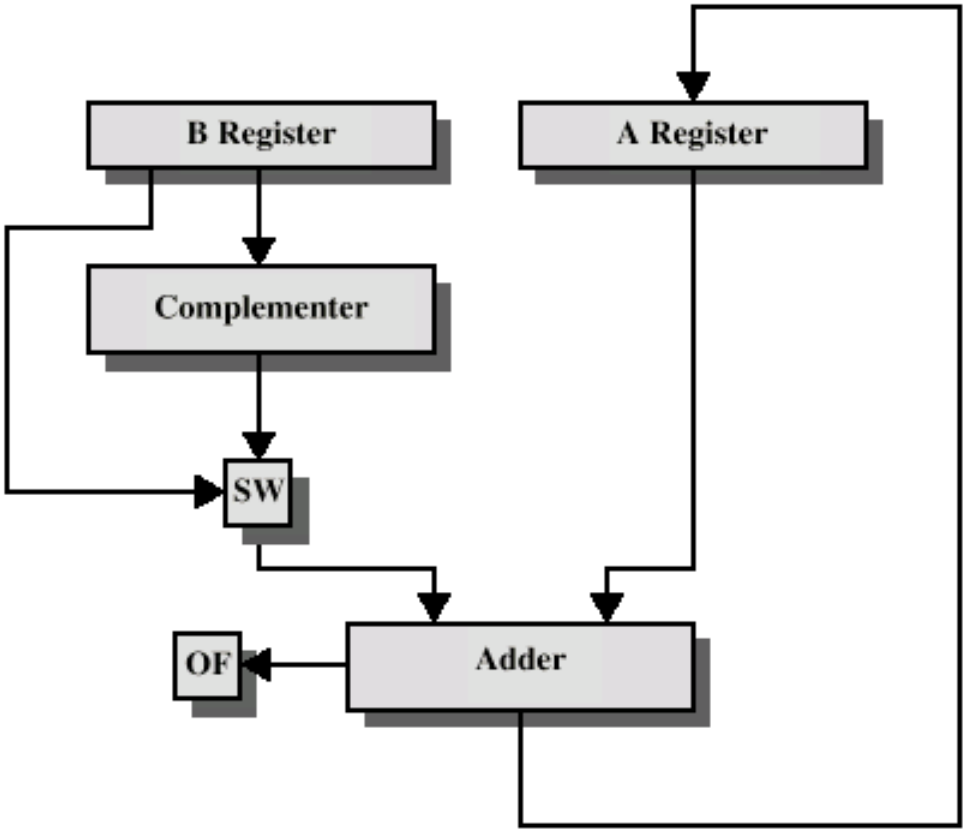
DIRECCIONAMIENTO RELATIVO

- **Direccionamiento por desplazamiento o dirección relativa**
  - Se utilizan dos campos de dirección en la instrucción
    - Uno es una referencia explicita a una dirección (A)
    - El otro es una referencia a un registro (R)
    - La dirección efectiva es EA = A + [R]
  - Direccionamiento relativo
    - A es agregado al Program Counter para causar una bifurcación antes de la búsqueda de la próxima instrucción
  - Direccionamiento por registro base
    - A es un desplazamiento agregado al contenido de un registro base implícito
    - Se utiliza por los programadores y el Sistema Operativo para identificar el comienzo de áreas de usuario, segmentos, etc.

FORMATOS DE INSTRUCCION

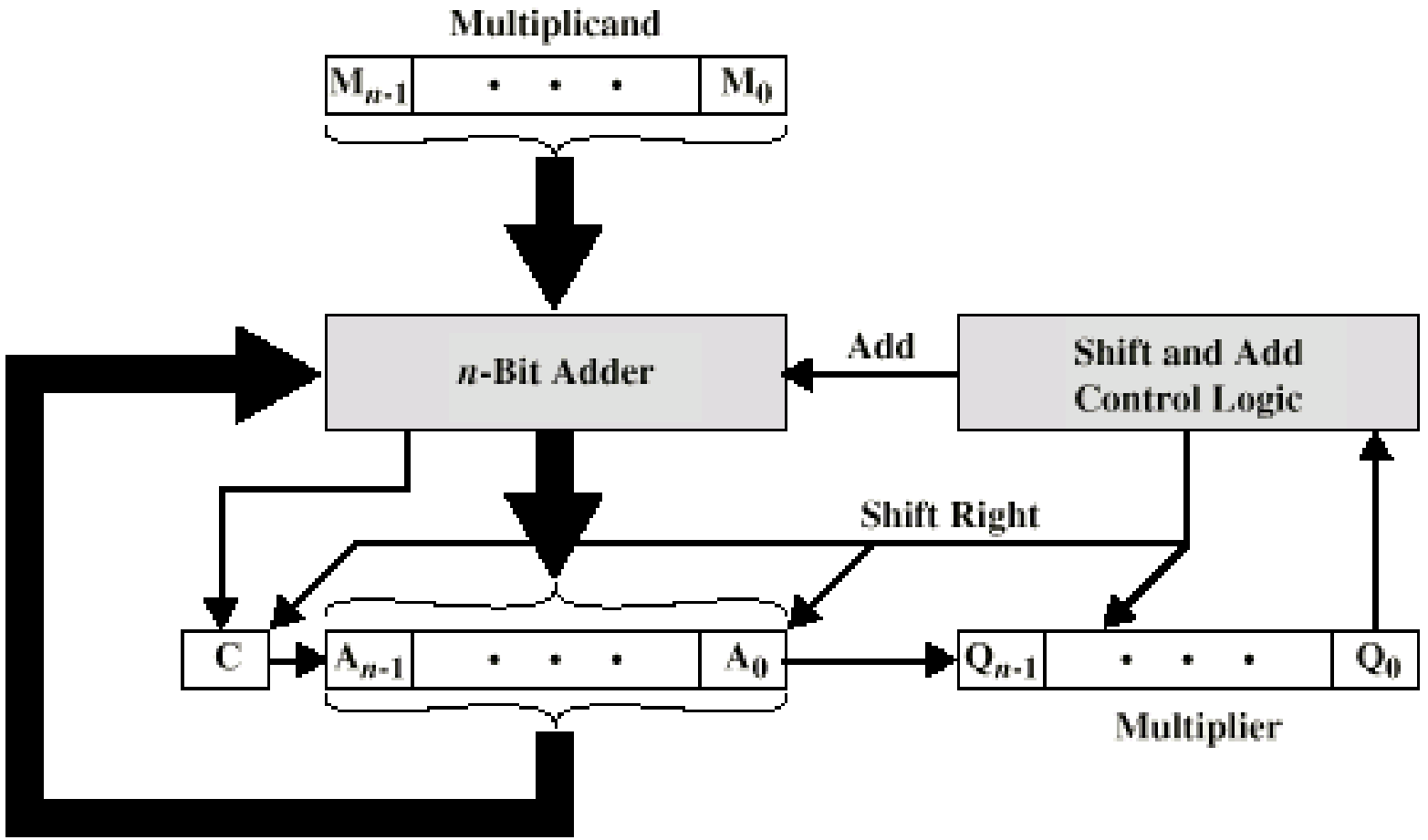
- El formato de instrucción define la distribución de las palabras de instrucción en términos de los elementos que la constituyen
- Una de las cuestiones básicas es la longitud de la instrucción
  - Instrucciones más largas permiten mayor número de códigos de operación, modos de direccionamiento, rangos de direccionamiento, etc.
  - No obstante, una mayor longitud no implica un aumento en la funcionalidad
  - La longitud de la instrucción es igual al tamaño de los datos que se transfieren entre la memoria y la CPU, o un múltiplo de ella
  - Si el sistema de memoria transfiere 32 bits, las instrucciones serán de 32 o 64 bits
- Asignación de bits
  - Para una determinada longitud de instrucción, implica un compromiso entre el número de códigos de operación soportados y la potencia de la capacidad de direccionamiento

HARDWARE PARA SUMAR Y RESTAR



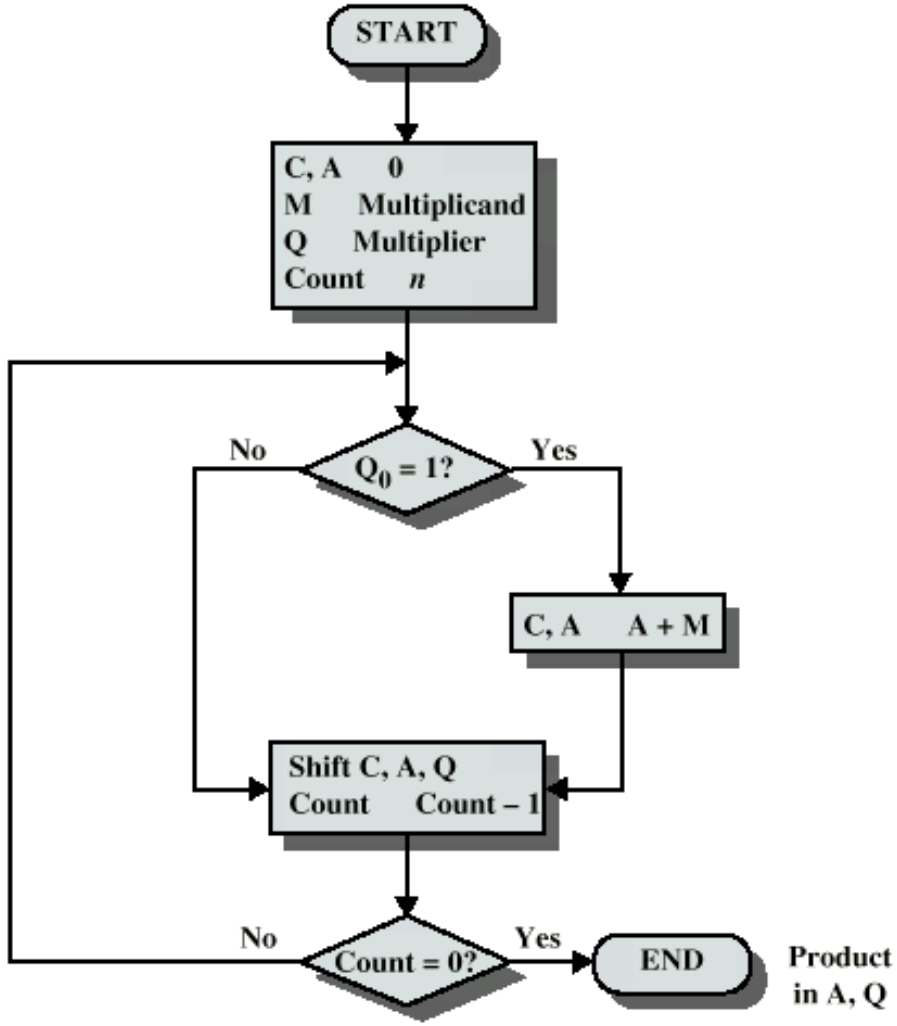
OF = overflow bit
SW = Switch (select addition or subtraction)

HARDWARE PARA MULTIPLICAR ENTEROS SIN SIGNO



(a) Block Diagram

MULTIPLICACION DE ENTEROS SIN SIGNO

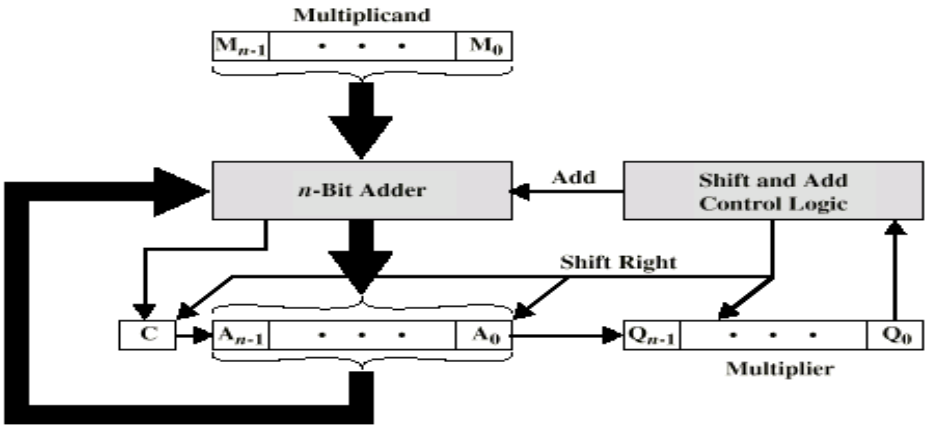


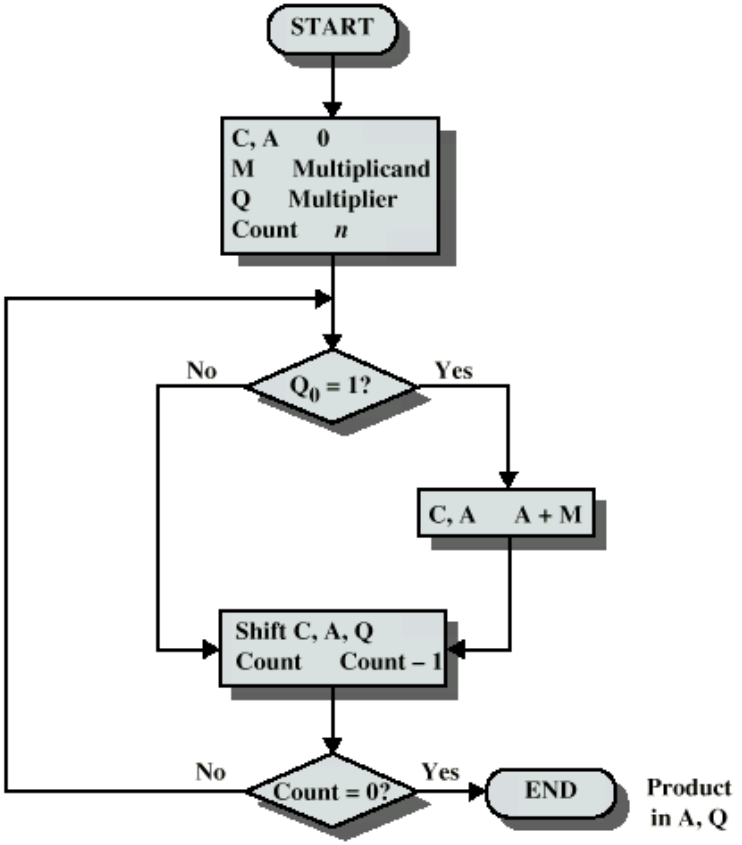
MULTIPLICACION DE ENTEROS SIN SIGNO

C	A	Q	M		
0	0000	1101	1011	Initial Values	
0	1011	1101	1011	Add	First Cycle
0	0101	1110	1011	Shift	
0	0010	1111	1011	Shift	Second Cycle
0	1101	1111	1011	Add	Third Cycle
0	0110	1111	1011	Shift	
1	0001	1111	1011	Add	Fourth Cycle
0	1000	1111	1011	Shift	

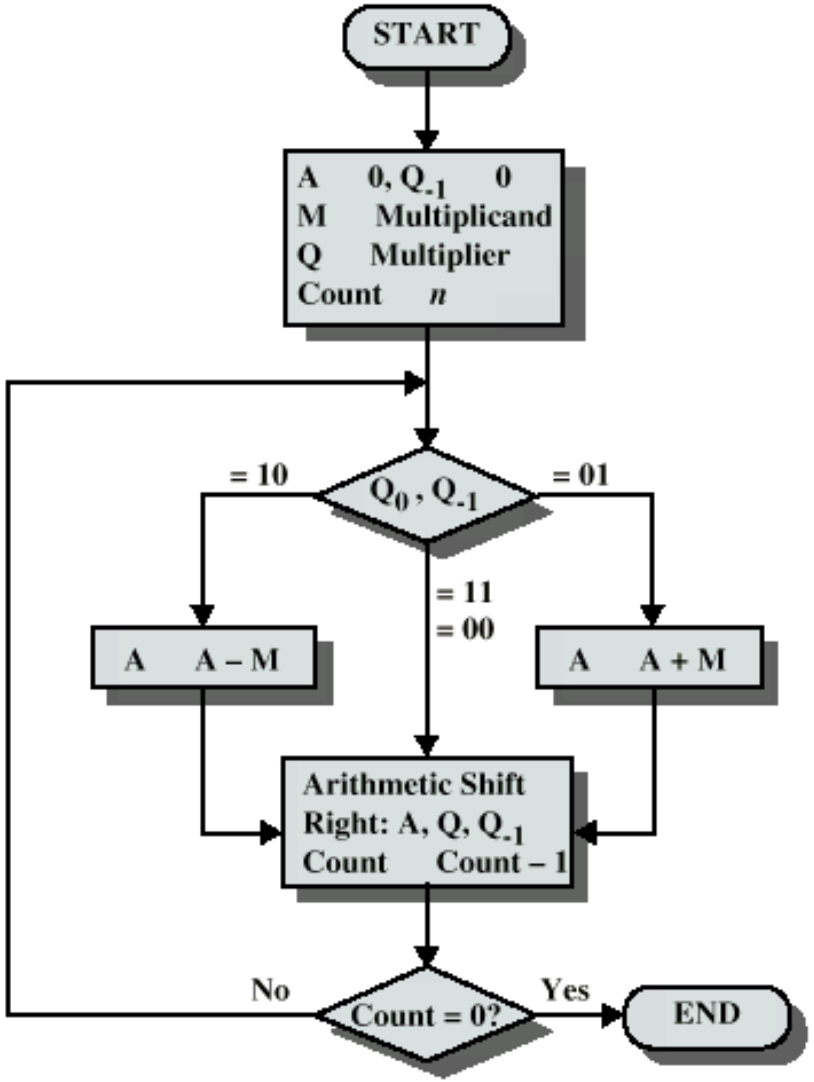
MULTIPLICACION DE ENTEROS SIN SIGNO

C	A	Q	M		
0	0000	1101	1011	Initial Values	
0	1011	1101	1011	Add	First Cycle
0	0101	1110	1011	Shift	
0	0010	1111	1011	Shift	Second Cycle
0	1101	1111	1011	Add	Third Cycle
0	0110	1111	1011	Shift	
1	0001	1111	1011	Add	Fourth Cycle
0	1000	1111	1011	Shift	





MULTIPLICACION DE ENTEROS CON SIGNO (ALGORITMO DE BOOTH)



MULTIPLICACION DE ENTEROS CON SIGNO

A	Q	Q_{-1}	M		
0000	0011	0	0111	Initial Values	
1001	0011	0	0111	A A - M	First Cycle
1100	1001	1	0111	Shift	
1110	0100	1	0111	Shift	Second Cycle
0101	0100	1	0111	A A + M	Third Cycle
0010	1010	0	0111	Shift	
0001	0101	0	0111	Shift	Fourth Cycle

DIVISION DE ENTEROS SIN SIGNO