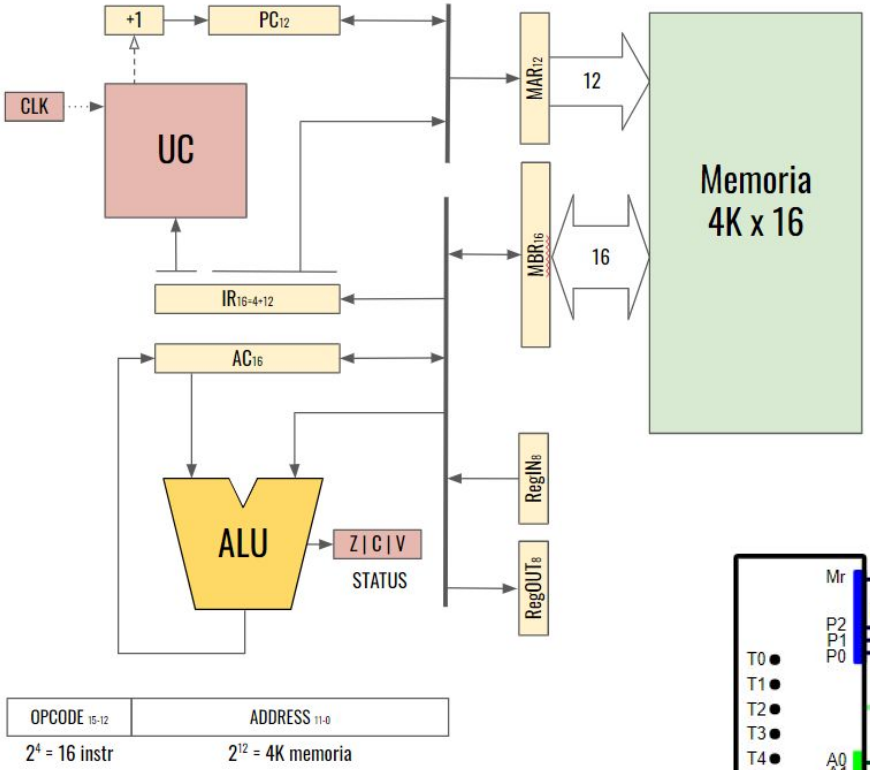
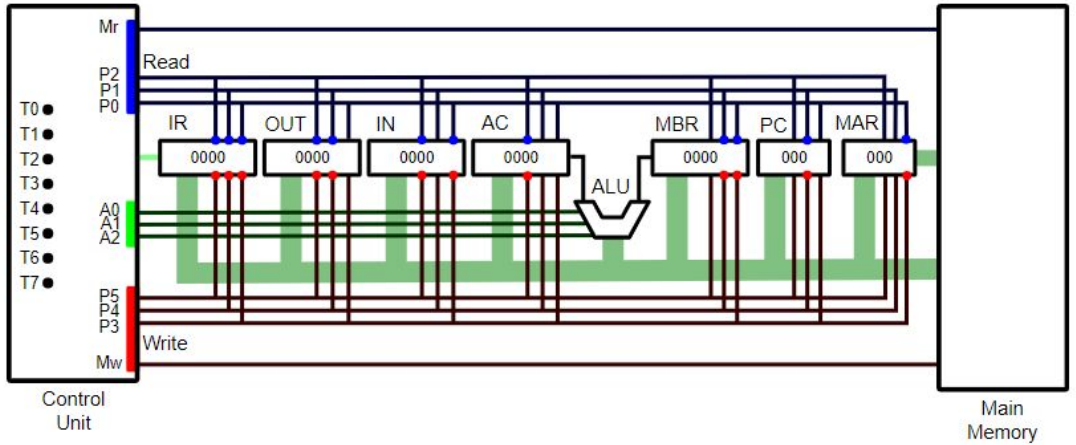
CLASE 4

MEJORANDO EL REPERTORIO DE INSTRUCCIONES (ISA)
Parte I
Repeticiones y nuevos modos de direccionamiento



http://marie.js.org

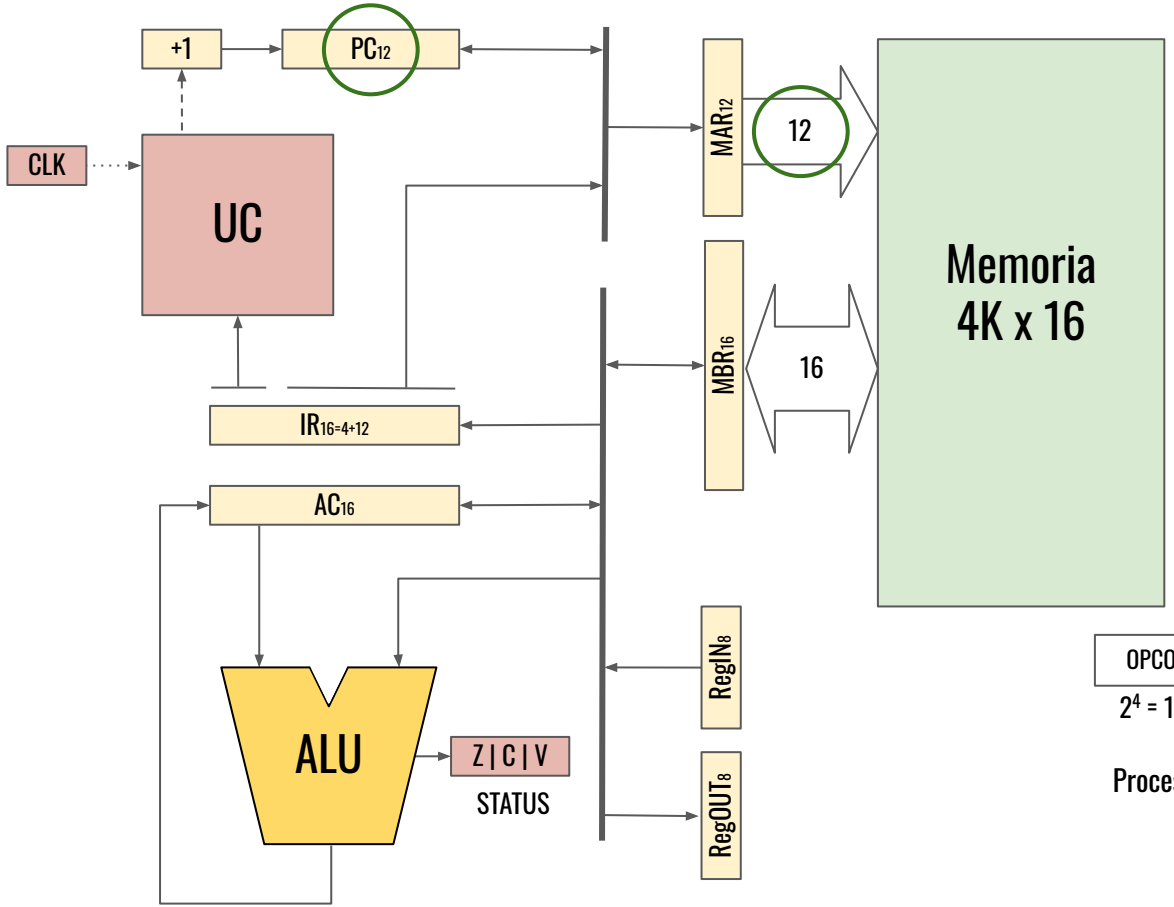


MEJORANDO EL REPERTORIO DE INSTRUCCIONES (ISA)

Algunas mejoras que suelen incorporarse a los procesadores. Requieren nuevas instrucciones y registros.
Mejora implica que se requieren menos instrucciones para realizar la misma tarea.
Pero ojo: no es tan importante el número de instrucciones como el tiempo total...
DISCUTIR EL IMPACTO EN LA ORGANIZACIÓN

a) Aumento de la cantidad de memoria
b) Aumento del número de registros de propósitos generales: disminución de los accesos a memoria.
c) Mejora de la ALU: operaciones y tipos de datos
d) Ampliación del repertorio de saltos condicionales
e) Loops: mejoras en las repeticiones
f) Nuevos modos de direccionamiento y registros de uso específico: mejora en el acceso a los datos
g) La pila: mejora en la implementación de subrutinas
h) El sistema de entrada/salida, interrupciones

Registro único
MARIE



OPCODE 15-12	ADDRESS 11-0
2^4 = 16 instr	2^{12} = 4K memoria

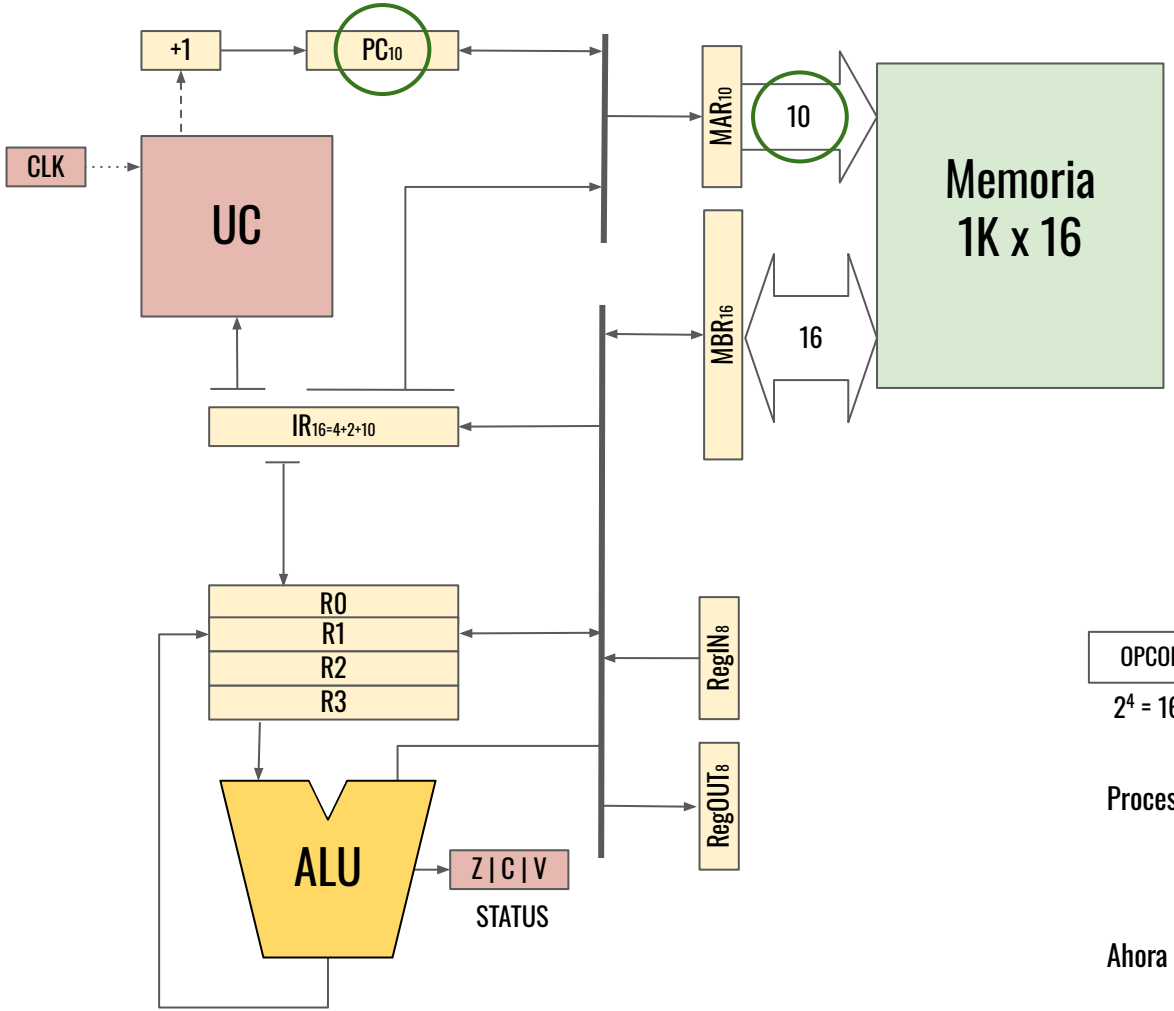
Procesador tipo (1,1)

```
LOAD X
ADD Y
STORE Z    / z=x+y
```

Clasificación (m,n)
m: número de operandos en memoria
n: número de operandos explícitos

Cuanta más memoria pueda manejar la arquitectura, mejor. Después veo cuánto pongo en una determinada implementación. Pero tengo que poder manejarla…

Banco de Registros
MARIE-R



Compromiso con la cantidad de memoria que es accesible ("direccionable")

OPCODE 15-12	R 11-10	ADDRESS 9-0
2^4 = 16 instr	2^2 = 4 reg	2^{10} = 1K memoria

Procesador tipo (1,2)

```
LOAD R1,X
ADD R1,Y
STORE Z,R1      / z=x+y
```

Ahora pueden implementarse operaciones entre registros

```
ADD R1,R2
```

```
for(int i=5; i>0; i--) {
      ...
}
```

```
i=0;
while(i>0) {
      ...
      i--;
}
```

```
i=0;
do {  ...
      i--;
} while(i>0);
```

compilador


```
/ Iteración MARIE

Loop,  ...      / bloque que repite
       ...      / utiliza el acumulador
       ...
       Load  Cont
       Subt  Step
       Store Cont
       Skipcond 400
       Jump  Loop
       Halt

Cont,  Dec 5
Step,  Dec 1
```

```
for(int i=5; i>0; i--) {
      ...
}
```

```
/ Iteración MARIE

Loop,  ...      / bloque que repite
       ...      / utiliza el acumulador
       ...
       Load  Cont
       Subt  Step
       Store Cont
       Skipcond 400
       Jump  Loop
       Halt

Cont,  Dec 5
Step,  Dec 1
```

La variable **Cont** podría almacenarse en un registro, por ejemplo **R3**, para no tener que ir y venir de memoria: **Subt R3, Step** o **Decr R3**

Podrían mejorarse los saltos, por ejemplo:**Jnz R3,Loop**

```
       Load R3,Cont
Loop,  ...      / bloque que repite
       ...
       Decr R3
       Jnz R3,Loop
       Halt
```

```
for(int i=5; i>0; i--) {
      ...
}
```

```
/ Iteración MARIE

Loop,  ...      / bloque que repite
       ...
       ...
       Load  Cont
       Subt  Step
       Store Cont
       Skipcond 400
       Jump  Loop
       Halt

Cont,  Dec 5
Step,  Dec 1
```

```
/ Iteración MARIE-R

       Load R3,Cont

Loop,  ...      / bloque que repite
       ...
       ...
       Decr R3
       Jnz R3,Loop
       Halt

Cont,  Dec 5
```

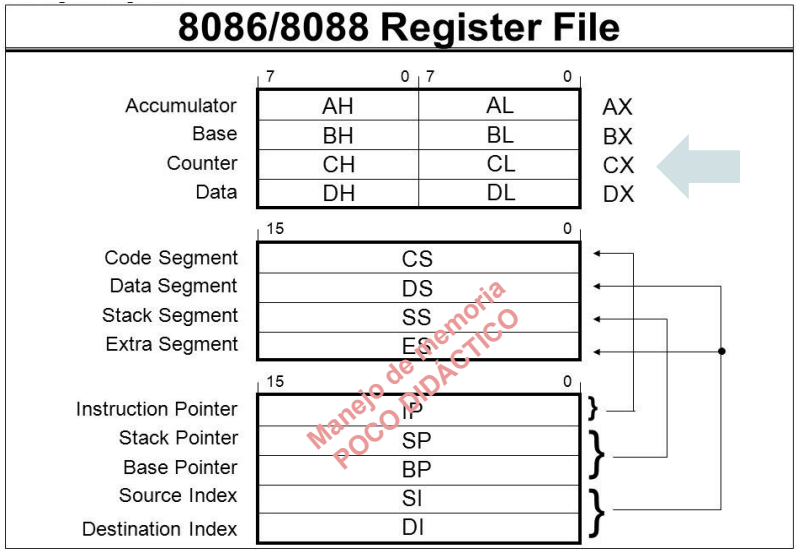
MEJORANDO EL REPERTORIO DE INSTRUCCIONES (ISA)

Algunas mejoras que suelen incorporarse a los procesadores. Requieren nuevas instrucciones y registros.
Mejora implica que se requieren menos instrucciones para realizar la misma tarea.
Pero ojo: no es tan importante el número de instrucciones como el tiempo total...
DISCUTIR EL IMPACTO EN LA ORGANIZACIÓN

a) Aumento de la cantidad de memoria
b) Aumento del número de registros de propósitos generales: disminución de los accesos a memoria.
c) Mejora de la ALU: operaciones y tipos de datos
d) Ampliación del repertorio de saltos condicionales
e) Loops: mejoras en las repeticiones
f) Nuevos modos de direccionamiento y registros de uso específico: mejora en el acceso a los datos
g) La pila: mejora en la implementación de subrutinas
h) El sistema de entrada/salida, interrupciones

Repeticiones en la arquitectura x86

x86: arquitectura tipo(1,2), acc (A) y varios registros especìficos de 16 bits, inst. de largo variable, opcode de 8-16 bits



Repetición en **MARIE**

```
/ Iteración MARIE

Loop,  ...      / bloque que repite
       ...
       ...
       Load  Cont
       Subt  Step
       Store Cont
       Skipcond 400
       Jump  Loop
       Halt

Cont,  Dec 5
Step,  Dec 1
```

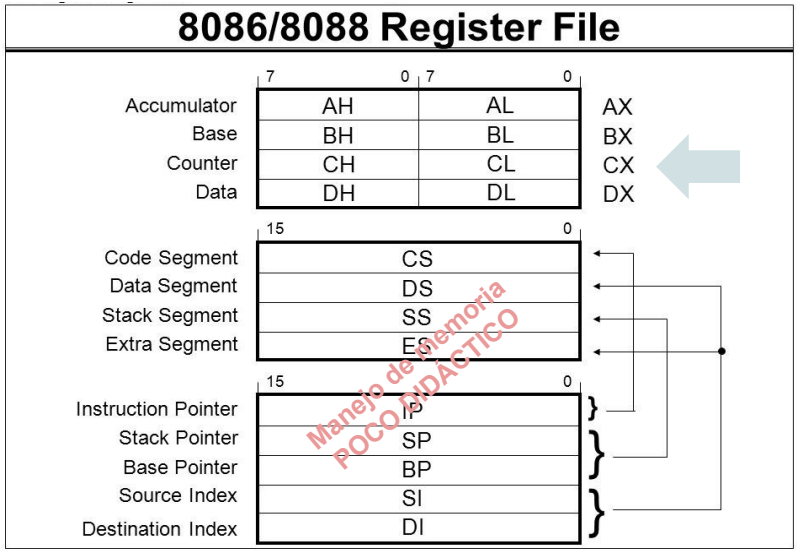
x86: una nueva instrucción LOOP y un registro CX. La instrucciòn automàticamente decrementa el registro CX y sólo salta si CX != 0

```
/ Iteración x86

mov  cx, 5
miloop: ...     / bloque que repite
        ...
        ...
        loop miloop
```

Repeticiones en la arquitectura x86

x86: arquitectura tipo(1,2), acc (A) y varios registros especìficos de 16 bits, inst. de largo variable, opcode de 8-16 bits



Repetición en **MARIE**

```
/ Iteración MARIE

Loop,  ...      / bloque que repite
       ...
       ...
       Load  Cont
       Subt  Step
       Store Cont
       Skipcond 400
       Jump  Loop
       Halt

Cont,  Dec 5
Step,  Dec 1
```

x86: una nueva instrucción LOOP y un registro CX. La instrucciòn automàticamente decrementa el registro CX y sólo salta si CX != 0

```
/ x86 PERFORMANCE - VER!
mov  cx, 5
miloop: ...     / bloque que repite
        ...
        ...
        dec cx
        jnz miloop
```

Interesante

Discutir ocupación de memoria y performance

```
/ Iteración x86

mov  cx, 5
miloop: ...     / bloque que repite
        ...
        ...
        loop miloop
```

```
for(int i=5; i>0; i++) {
      ...
}
```

```
/ Iteración MARIE

Loop,  ...      / bloque que repite
       ...
       ...
       Load  Cont
       Subt  Step
       Store Cont
       Skipcond 400
       Jump  Loop
       Halt

Cont,  Dec 5
Step,  Dec 1
```

```
/ Iteración MARIE-R

       Load R3,Cont

Loop,  ...      / bloque que repite
       ...
       ...
       Dec R3
       Jnz R3,Loop
       Halt

Cont,  Dec 5
```

```
/ Iteración MARIE-R con repetición

       Load R3,Cont

Loop,  ...      / bloque que repite
       ...
       ...
       Repeat Loop
       Halt

Cont,  Dec 5
```

Notar el exceso de instrucciones debido a la repetición en cada caso: 3+5, 3+2 y 3+1.

Mejora = 8/4 = 2

En principio se obtendría el doble de performance, si todas las instrucciones tardaran lo mismo.
Pero, ¿cuántos ciclos tomaría una eventual instrucción Repeat en MARIE? Proponer una y hacer bien la cuenta.

Intervalo 15’

MEJORANDO EL REPERTORIO DE INSTRUCCIONES (ISA)

Algunas mejoras que suelen incorporarse a los procesadores. Requieren nuevas instrucciones y registros.
Mejora implica que se requieren menos instrucciones para realizar la misma tarea.
Pero ojo: no es tan importante el número de instrucciones como el tiempo total...
DISCUTIR EL IMPACTO EN LA ORGANIZACIÓN

a) Aumento de la cantidad de memoria
b) Aumento del número de registros de propósitos generales: disminución de los accesos a memoria.
c) Mejora de la ALU: operaciones y tipos de datos
d) Ampliación del repertorio de saltos condicionales
e) Loops: mejoras en las repeticiones
f) Nuevos modos de direccionamiento y registros de uso específico: mejora en el acceso a los datos
g) La pila: mejora en la implementación de subrutinas
h) El sistema de entrada/salida, interrupciones

Modos de direccionamiento de MARIE

(1,1), Acc 16 bits, inst. de largo fijo de 16 bits, opcode de 4 bits
Punteros, direccionamiento indirecto

Add X direccionamiento directo (OPC=0010)

```
MAR ← X
MBR ← M[MAR]
AC  ← AC + MBR
```

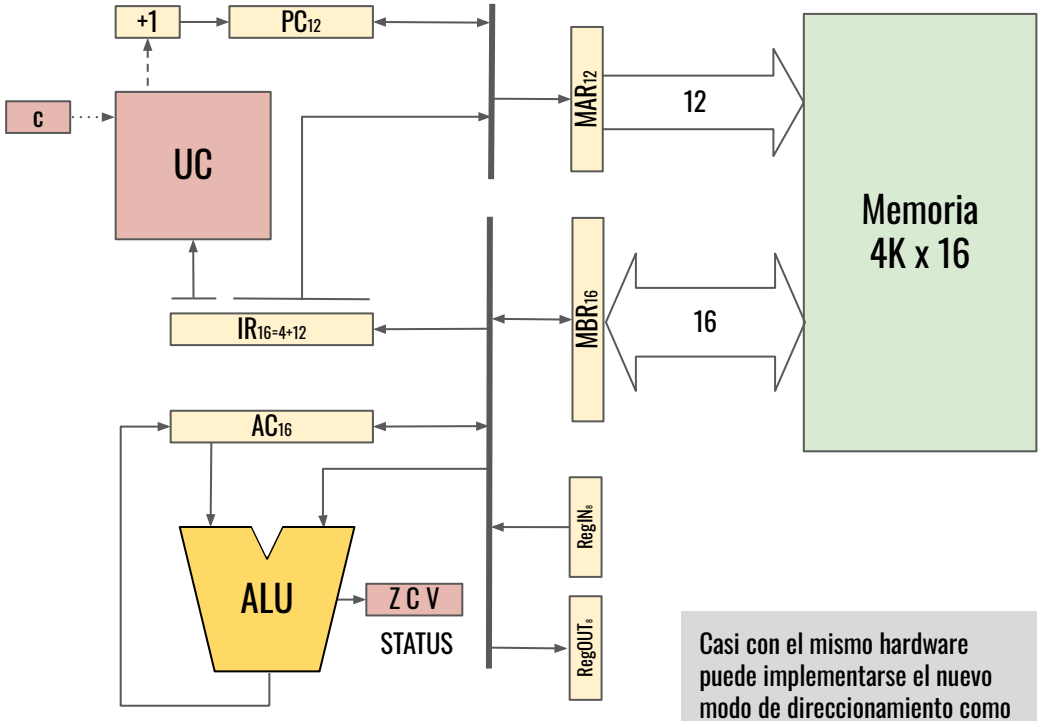
AddI X direccionamiento indirecto (OPC=1011) **Add (X)**

```
MAR ← X
MBR ← M[MAR]
MAR ← MBR
MBR ← M[MAR]
AC  ← AC + MBR
```

Clear direccionamiento implícito (OPC=1010)

```
AC  ← 0
```

OPC 15-12	X 11-0



Modo de direccionamiento:
Es la forma en que se especifica la dirección efectiva del OPERANDO de la instrucción.
Son una característica muy importante del diseño de un repertorio de instrucciones (ISA).

En la codificación, qué significa X?
En Add es la dirección del argumento
En AddI es la dirección de la dirección

Casi con el mismo hardware puede implementarse el nuevo modo de direccionamiento como una secuencia diferente de la UC (microcódigo).

Otros modos de direccionamiento

que podrían agregarse a MARIE

Add X direccionamiento directo

```
MAR ← X
MBR ← M[MAR]
AC  ← AC + MBR
```

AddI X direccionamiento indirecto

```
MAR ← X
MBR ← M[MAR]
MAR ← MBR
MBR ← M[MAR]
AC  ← AC + MBR
```

Clear direccionamiento implícito

```
AC  ← 0
```

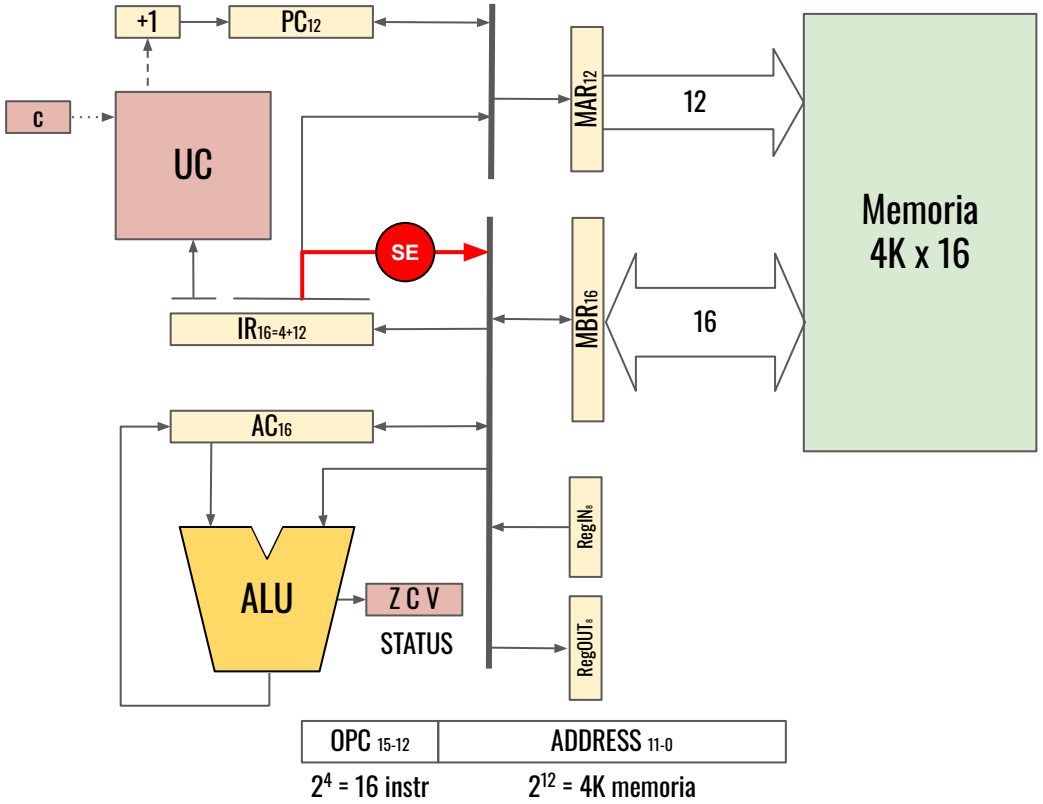
NUEVO

AddM K direccionamiento inmediato

```
MBR ← K
AC  ← AC + MBR
```

CONSTANTES
Limitación 12 bits
SIGN EXTENSION (SE)

OPC 15-12	K 11-0



Otros modos de direccionamiento

que podrían agregarse a MARIE

Add X direccionamiento directo

```
MAR ← X
MBR ← M[MAR]
AC  ← AC + MBR
```

AddI X direccionamiento indirecto

```
MAR ← X
MBR ← M[MAR]
MAR ← MBR
MBR ← M[MAR]
AC  ← AC + MBR
```

Clear direccionamiento implícito

```
AC  ← 0
```

NUEVO

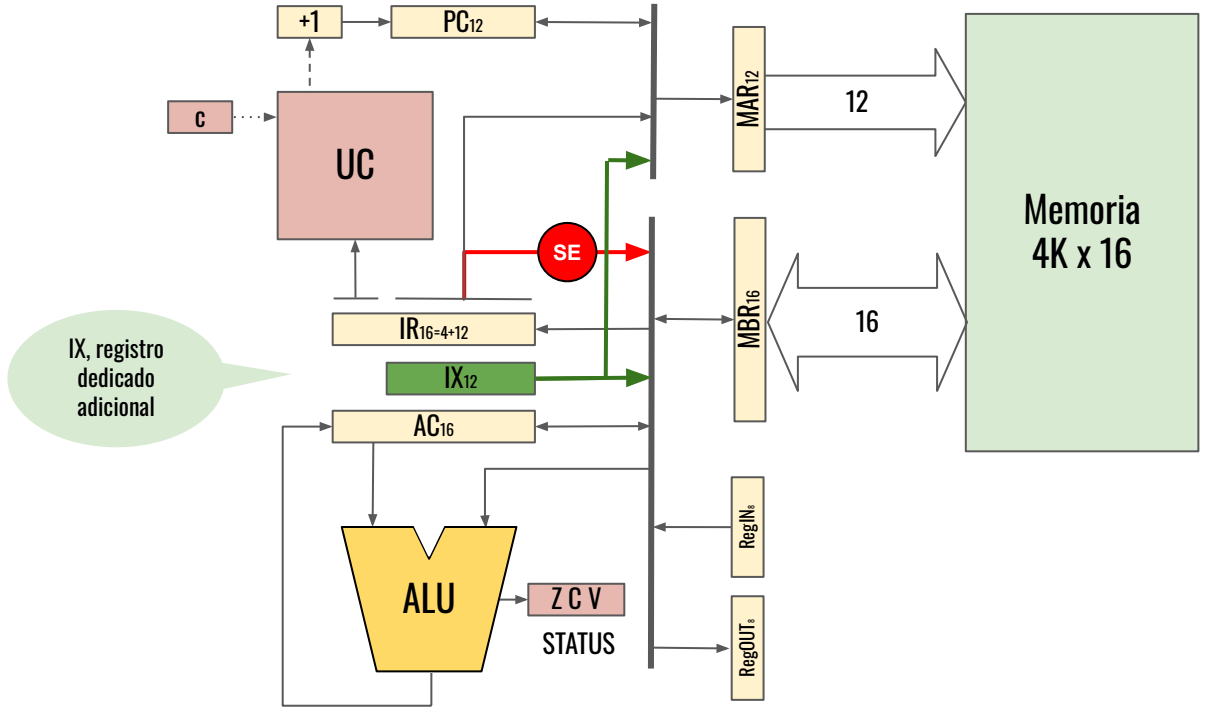
AddM K direccionamiento inmediato

```
MBR ← K
AC  ← AC + MBR
```

NUEVO

AddX Off direccionamiento indexado

```
MBR ← IX
AC  ← Off
AC  ← AC + MBR
MAR ← AC
MBR ← M[MAR]
AC  ← AC + MBR
```



Otros modos de direccionamiento

que podrían agregarse a MARIE

Add X direccionamiento directo

```
MAR ← X
MBR ← M[MAR]
AC  ← AC + MBR
```

AddI X direccionamiento indirecto

```
MAR ← X
MBR ← M[MAR]
MAR ← MBR
MBR ← M[MAR]
AC  ← AC + MBR
```

Clear direccionamiento implícito

```
AC  ← 0
```

NUEVO

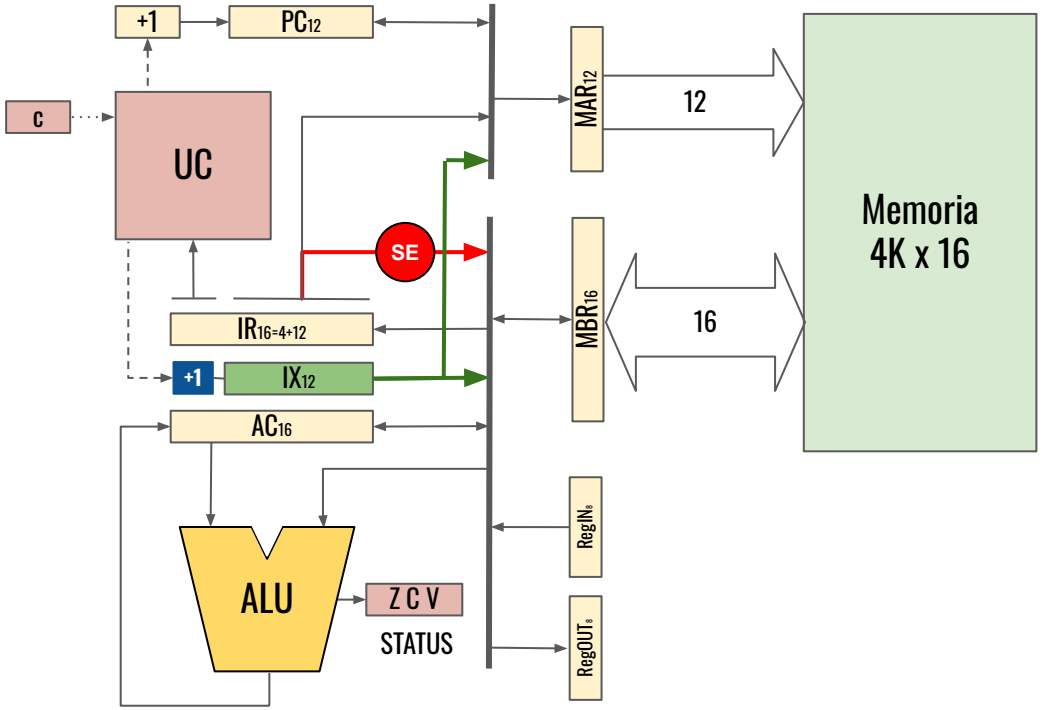
AddM K direccionamiento inmediato

```
MBR ← K
AC  ← AC + MBR
```

NUEVO

AddX+ Off indexado con post-incremento

```
MBR ← IX
IX ← IX + 1
AC  ← Off  ;
AC  ← AC + MBR
MAR ← AC
MBR ← M[MAR]
AC  ← AC + MBR
```



OPC 15-12	Offset 11-0

```
        / Hipotético
        Cargar loop reg
        Cargar index reg
Loop,   AddX+
        Repeat Loop
        Store Sum
        Halt
```

$$t = \frac{N \times CPI}{f_{clock}}$$

Dependiendo de la implementación, la frecuencia podría disminuir

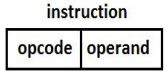
```
/ MARIE
        Load  Addr        /Load address of first number to be added
        Store Next        /Store this address is our Next pointer
        Load  Num         /Load the number of items to be added
        Subt  One         /Decrement
        Store Cont        /Store this value in Ctr to control looping
Loop,   Load  Sum         /Load the Sum into AC
        AddI  Next        /Add the value pointed to by location Next
        Store Sum         /Store this sum
        Load  Next        /Load Next
        Add   One         /Increment by one to point to next address
        Store Next        /Store in our pointer Next
        Load  Cont        /Load the loop control variable
        Subt  One         /Subtract one from the loop control variable
        Store Cont        /Store this new value in loop control variable
        Skipcond 000      /If control variable < 0, skip next instruction
        Jump Loop         /Otherwise, go to Loop
        Halt              /Terminate program
Addr,   Hex 117           /Numbers to be summed start at location 0x118
Next,   Hex 0             /A pointer to the next number to add
Num,    Dec 5             /The number of values to add
Sum,    Dec 0             /The sum
Cont,   Hex 0             /The loop control variable
One,    Dec 1             /Used to increment and decrement by 1
        Dec 10            /The values to be added together
        Dec 15
        Dec 20
        Dec 25
        Dec 30
```

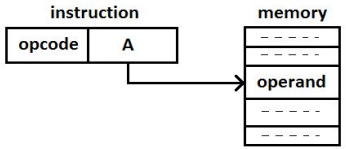
Modos de direccionamiento

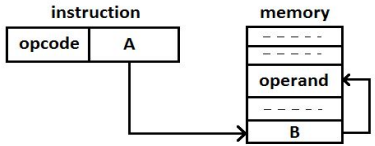
Definición: Identificación de la ubicación (dirección efectiva) del o los operandos de una instrucción
Objetivos: Programación eficiente, código compacto y estructuras de datos complejas
Reducción del número de bits del campo operando

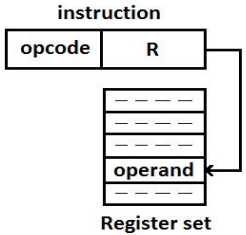
0. Implícito o inherente
1. Inmediato (constantes)
2. Directo
3. Indirecto
4. De registro (directo)
5. Indirecto por registro (indexado)
6. Desplazamiento (indirecto vía registro con offset)

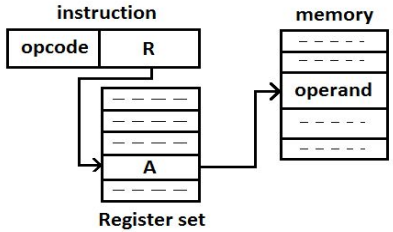
Los nombres pueden variar entre arquitecturas

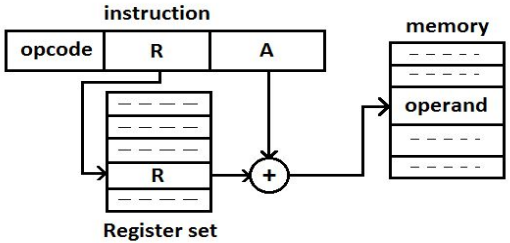








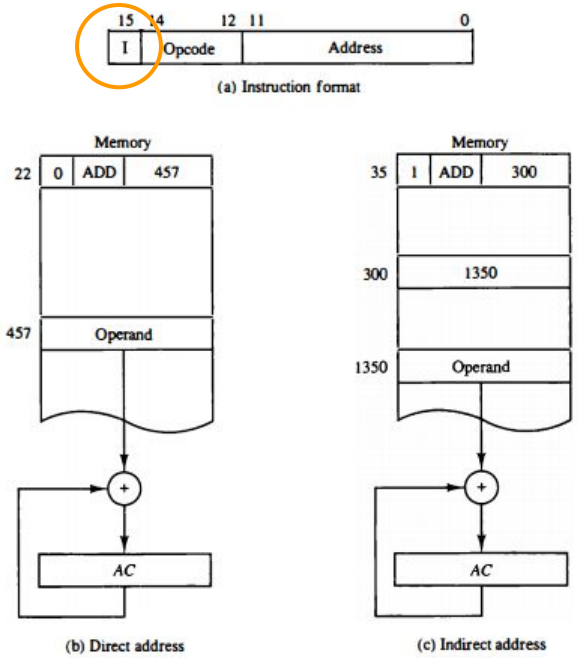




https://www.infor.uva.es/~bastida/OC/modos.pdf

Modos de direccionamiento

Codificación: **son instrucciones diferentes**




```
Add M
```

```
Add (M)
AddI M
```

La sintaxis en **lenguaje ensamblador** puede variar

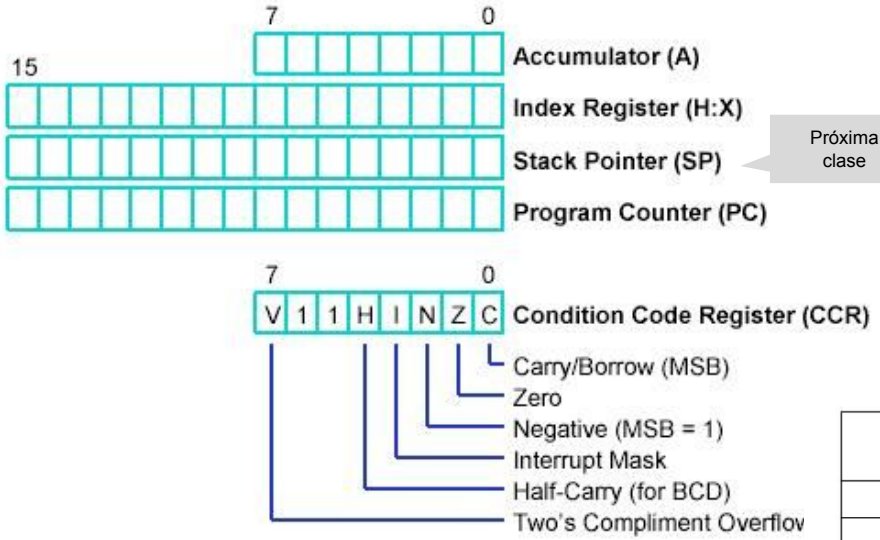
Ortogonalidad del Repertorio de Instrucciones

Todos los tipos de instrucciones pueden utilizar todos los modos de direccionamiento (se dice ortogonal porque el tipo y el modo son independientes. Bits de modo en la codificación. No es el caso de MARIE

Disponer de un ISA ortogonal solía considerarse una característica muy deseable hasta la irrupción de los diseños RISC de los 90s. Lo discutiremos en profundidad más adelante.

Modos de direccionamiento en la arquitectura CPU08

Motorola/Freescale/NXP 68HC08, tipo(1,1), Acc 8 bits y registros especiales, inst. de largo variable, opcode=8 bits




```
ADD #$01        ; inmediato #
ADD $0F         ; directo
ADD $010F       ; directo extendido
ADD ,X          ; indexado
ADD $02,X       ; indexado con offset
ADD $10FF,X
```

NXP MC68HC908QT1CPE
32MHz, 1.5 kB Flash, 8-Pin PDIP

Source Forms	Addr Mode	Machine Code		HC08 Cycles
		Opcode	Operand(s)	
ADD #*opr*	IMM	AB	ii	2
ADD *opr*	DIR	BB	dd	3
ADD *opr*	EXT	CB	hh ll	4
ADD ,X	IX	FB		2
ADD *opr*,X	IX1	EB	ff	3
ADD *opr*,X	IX2	DB	ee ff	4
ADD *opr*,SP	SP1	9EEB	ff	4
ADD *opr*,SP	SP2	9EDB	ee ff	5

1974 - 50 años

Microprocesadores con datos de 8 bits (ALU, Acc)
y direcciones de 16 bits (SP, IX)

Motorola 6800

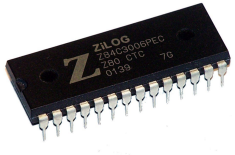
(sixty eight hundred)

Intel 8080

(eighty eighty)

Zilog Z80

(zed eighty)



MEJORANDO EL REPERTORIO DE INSTRUCCIONES (ISA)

Algunas mejoras que suelen incorporarse a los procesadores. Requieren nuevas instrucciones y registros.
Mejora implica que se requieren menos instrucciones para realizar la misma tarea.
Pero ojo: no es tan importante el número de instrucciones como el tiempo total...
DISCUTIR EL IMPACTO EN LA ORGANIZACIÓN

a) Aumento de la cantidad de memoria
b) Aumento del número de registros de propósitos generales: disminución de los accesos a memoria.
c) Mejora de la ALU: operaciones y tipos de datos
d) Ampliación del repertorio de saltos condicionales
e) Loops: mejoras en las repeticiones
f) Nuevos modos de direccionamiento y registros de uso específico: mejora en el acceso a los datos
g) La pila: mejora en la implementación de subrutinas
h) El sistema de entrada/salida, interrupciones

Próxima clase