

Seminario de Solución de problemas de Traductores de Lenguajes I

Centro Universitario de Ciencias Exactas en
ingenierías

Universidad de Guadalajara



Maestro: Tonatiuh Hernandez Casas

Juan Antonio Pérez Juárez
Código: 215660996
Carrera: INCO

Actividad 1: Parte I

Microprocesadores Intel 80x86

Descripción de la actividad:

Elabore una tabla comparativa de la familia de microprocesadores Intel 80x86 en orden cronológico, presentando características como la velocidad, el tamaño de los registros, capacidad de memoria, registros y buses de datos.

Desarrollo de la actividad:

Creo que lo primero para hablar de este tema es definir lo que es un microprocesador.

El procesador o microprocesador es la unidad de procesamiento principal de un ordenador, es por ello la unidad más importante, el «cerebro» de un ordenador.

Es el encargado de ejecutar todos los programas, desde el sistema operativo hasta las aplicaciones de usuario; sólo ejecuta instrucciones en lenguaje máquina, realizando operaciones aritméticas y lógicas simples, tales como sumar, restar, multiplicar, dividir y las operaciones bitwise, también las lógicas binarias y accesos a memoria.

Puede contener una o más unidades centrales de procesamiento (CPU) constituidas, esencialmente, por registros, una unidad de control, una unidad aritmético lógica (ALU) y una unidad de cálculo en coma flotante (FPU) (conocida antiguamente como «coprocesador matemático»).

El microprocesador está conectado generalmente mediante un zócalo específico de la placa base de la computadora; normalmente para su correcto y estable funcionamiento, se le incorpora un sistema de refrigeración que consta de un disipador de calor, fabricado de algún material de alta conductividad térmica, como cobre o aluminio, y de uno o más ventiladores que eliminan el exceso del calor absorbido por el disipador. Entre el disipador y la cápsula del microprocesador usualmente se coloca pasta térmica para mejorar la conductividad del calor. Existen otros métodos más eficaces, como la refrigeración líquida o el uso de células peltier para refrigeración extrema, pero estas técnicas se utilizan casi exclusivamente para aplicaciones especiales, tales como en las prácticas de overclocking.

Dato Curioso:

En 1993 apareció el Pentium. No se siguió con la nomenclatura 80586 porque muchas empresas competidoras de Intel habían comenzado a producir CPUs con el mismo número que los de Intel. Ante el hecho de que un número no puede ser usado como marca registrada, los procesadores llevan un nombre propio.

Una pequeña línea del tiempo:

Una pequeña línea del tiempo.

- 1978 y 1979 Intel 8086 y 8088. Primeros microprocesadores de la Arquitectura x86.

- 1980 Intel 8087. Primer coprocesador numérico de la arquitectura x86, inicio de la serie x87.
- 1980 NEC V20 y V30. Clones de procesadores 8088 y 8086, respectivamente, fabricados por NEC.
- 1982 Intel 80186 y 80188. Mejoras del 8086 y 8088.
- 1982 Intel 80286. Aparece el modo protegido, tiene capacidad para multitarea.
- 1985 Intel 80386. Primer microprocesador x86 de 32 bits.
- 1989 Intel 80486. Incorpora el coprocesador numérico en el propio circuito integrado.
- 1993 Intel Pentium. Mejor desempeño, arquitectura superescalar.
- 1995 Pentium Pro. Ejecución fuera de orden y Ejecución especulativa
- 1996 Amd k5. Rival directo del Intel Pentium.
- 1997 Intel Pentium II. Mejora la velocidad en código de 16 Bits, incorpora MMX
- 1998 AMD K6-2. Competidor directo del Intel Pentium II, introducción de 3DNow!
- 1999 Intel Pentium III. Introducción de las instrucciones SSE
- 2000 Intel Pentium 4. NetBurst. Mejora en las instrucciones SSE
- 2005 Intel Pentium D. EM64T. Bit NX, Intel Viiv
- 2006 Intel Core 2. Introducción de microarquitectura Intel P8. Menor consumo, múltiples núcleos, soporte de virtualización en hardware incluyendo x86-64 y SSSE3.

Procesador	Detalles
8086 / 80186	<p>El 8086 dispone de instrucciones especiales para el tratamiento de cadenas de caracteres.</p> <p>Los registros del 8086 tienen una misión específica, por lo que se podría decir que cada uno de ellos tiene su propia personalidad, aunque varios comparten tareas comunes.</p> <p>El encapsulado del 8086 está formado por 40 patillas, simplificando así el hardware, aunque por contra, es necesario la multiplexación del bus de datos con el de direcciones.</p> <p>El 8086 dispone de un conjunto de registros, denominados 'cola de instrucciones', en el cual se van almacenando de forma anticipada los códigos de las instrucciones, consiguiendo que este aumente su velocidad de trabajo.</p> <p>Las 20 líneas del bus de direcciones sólo permiten direccionar una memoria de 1 Megabyte.</p> <p>El 8086 requiere una señal de reloj exterior, siendo 5</p>

	<p>y 8 Mhz las frecuencias típicas de funcionamiento. El 8086 dispone de una arquitectura “pipeline”, es decir, que la CPU puede seguir leyendo instrucciones.</p>
80286	<p>El i286 funciona el doble de rápido que su predecesor, el Intel 8086, y puede direccionar hasta 16 Mbytes de memoria RAM, en contraposición a 1 Mbyte Del i8086.</p> <p>El i286 fue diseñado para ejecutar aplicaciones multitarea, incluyendo comunicaciones, control de procesos en tiempo real y sistemas multiusuario.</p> <p>Otras características incluyen todas las características del juego de instrucciones del 80186, así como la extensión del espacio direccionable a 16 MB, utilizando 24 bits para direccionar ($2^{24} = 16.777.216$).</p>
80386	<p>El 80386 consiste en una unidad central de proceso (CPU), una unidad de manejo de memoria (MMU) y una unidad de interfaz con el bus (BIU).</p> <p>La CPU está compuesta por la unidad de ejecución y la unidad de instrucciones. La unidad de ejecución contiene los ocho registros de 32 bits de propósito general que se utilizan para el cálculo de direcciones y operaciones con datos y un barrel shifter de 64 bits que se utiliza para acelerar las operaciones de desplazamiento, rotación, multiplicación y división. Al contrario de los microprocesadores previos, la lógica de división y La multiplicación utiliza un algoritmo de 1 bit por ciclo de reloj.</p> <p>El algoritmo de multiplicación termina la iteración cuando los bits más significativos del multiplicador son todos ceros, lo que permite que las multiplicaciones típicas de 32 bits se realicen en menos de un microsegundo.</p> <p>La memoria se organiza en uno o más segmentos de longitud variable, con un tamaño máximo de 4 gigabytes.</p> <p>Estos segmentos, como se vio en la explicación del 80286, tienen atributos asociados, que incluyen su ubicación, tamaño, tipo (pila, código o datos) y características de protección.</p> <p>El 80386 tiene dos modos de operación: modo de direccionamiento real (modo real), y modo de direccionamiento virtual protegido (modo protegido).</p> <p>En modo real el 80386 opera como un 8086 muy rápido, con extensiones de 32 bits si se desea. El</p>

	<p>modo real se requiere primariamente para preparar el procesador para que opere en modo protegido. El modo protegido provee el Acceso al sofisticado manejo de memoria y paginado.</p> <p>Finalmente, para facilitar diseños de hardware de alto rendimiento, la interfaz con el bus del 80386 ofrece pipelining de direcciones, tamaño dinámico del ancho del bus de datos (puede tener 16 ó 32 bits según se desee en un determinado ciclo de bus) y señales de habilitación de bytes por cada byte del bus de datos. Hay más información sobre esto en la sección de hardware del 80386.</p>
86486	<p>Este microprocesador es básicamente un 80386 con el agregado de una unidad de punto flotante compatible con el 80387 y un caché de memoria de 8 KBytes. Por lo tanto los bloques que componen el 80486 son los siguientes:</p> <ol style="list-style-type: none"> 1. Unidad de Segmentación 2. Unidad de almacenamiento 3. Unidad de Caché 4. Interfaz con Bus 5. Unidad de Instrucciones 6. Unidad de Punto Flotante <p>En el caso del 80486, cada unidad de memoria son 16 bytes. Esta cantidad es una línea del caché. Las líneas pueden ser válidas (cuando contienen datos de la memoria principal) o inválidas (en este caso la línea no contiene información útil). Como el caché se llena por líneas completas (comenzando por direcciones múltiplos de 16), hay que tratar de no leer posiciones aleatorias de la memoria, ya que en este caso, si se leen bytes en posiciones alejadas unas de otras, el procesador usará cuatro ciclos de bus para leer 16 bytes (para llenar una línea) por cada byte que deseamos leer. Esto no es problema para el código o la pila (stack) ya que éstos se acceden generalmente de manera secuencial.</p> <p>El caché es una memoria especial, llamada memoria asociativa. Dicha memoria tiene, asociado a cada unidad de memoria, un tag, que almacena la dirección de memoria que contiene los datos que están en la unidad de memoria.</p> <p>Cuando se desea leer una posición de memoria mediante esta memoria asociativa, se comparan todos los tags con esta dirección. Si algún tag tiene esta dirección, se dice que hubo un acierto (cache hit en inglés) con lo que se puede leer la información asociada a ese tag. En caso contrario hay un fallo</p>

	<p>(cache miss en inglés), con lo que hay que perder un ciclo de bus para leer el dato que está en memoria externa.</p> <p>Hay dos clases de cache: write-through y write-back (retro escritura)(implementado solamente en los modelos write-back enhanced DX2 y write-back enhanced DX4). La diferencia entre las dos radica en el momento de escritura.</p> <p>Las primeras siempre escriben en la memoria principal, mientras que las otras sólo escriben cuando se llena el caché y hay que desocupar una línea. Esto último aumenta el rendimiento del sistema.</p>
Pentium (80586)	<p>El 19 de octubre de 1992, Intel anunció que la quinta generación de su línea de procesadores compatibles (cuyo código interno era el P5) llevaría el nombre Pentium en vez de 586 u 80586, como todo el mundo estaba esperando.</p> <p>Esta fue una estrategia de Intel para poder registrar la marca y así poder diferir el nombre de sus procesadores del de sus competidores (AMD y Cyrix principalmente).</p> <p>Lo que comenzó con la técnica del 386/486 de tener vías de acceso múltiples para la ejecución de instrucciones, se ve refinado en el Pentium ya que tiene un diseño con doble vía de acceso. El objetivo de ésta es el de procesar múltiples instrucciones simultáneamente, en varios estados de ejecución, para obtener una velocidad de ejecución general de instrucciones de una instrucción por ciclo de reloj.</p> <p>El resultado final de la estructura doble vía de acceso es un diseño superescalar que tiene la habilidad de ejecutar más de una instrucción en un ciclo de reloj dado. Los procesadores escalares, como la familia del 486, tienen sólo una vía de acceso.</p> <p>Se puede pensar que el microprocesador moderno con vías de acceso doble es similar a una línea de producción que recibe en un extremo materias primas sin procesar y a medio procesar y que saca el producto terminado en el otro extremo. La línea de producción con vía de acceso doble del Pentium transforma la materia prima de información y de código de software en el producto terminado. El Pentium sigue el modelo de vía de acceso del 486, ejecutando instrucciones simples con enteros en un ciclo de reloj. Sin embargo es más exacto decir que aquellas instrucciones estaban en la etapa de ejecución de la vía de acceso durante un ciclo de</p>

	<p>reloj. Siempre se requieren ciclos adicionales de reloj para buscar, decodificar la instrucción y otros procesos vitales. La secuencia de funcionamiento de la vía de datos es como sigue:</p> <p>prebúsqueda, decodificación 1, decodificación 2, ejecución y retro escritura.</p> <p>Los aspectos superescalares del Pentium dependen de su vía de acceso doble. Los procesadores superescalares permiten que se ejecute más de una instrucción por vez. El procesador tiene dos vías de acceso de enteros, una en forma de U y otra en forma de V y automáticamente aparea las instrucciones para incrementar la proporción de instrucciones por ciclo de reloj para que sea mayor que 1.</p> <p>Si el tener múltiples instrucciones pasando por dos vías suena como el equivalente de un tranque en el tráfico del microprocesador, eso no es así, porque hay reglas y restricciones que evitan las colisiones y los retrasos.</p>
--	---

Resultados de la actividad: No Aplica

Reflexión: Esta parte de la actividad me gustó mucho porque no sabía mucho de la historia de estos procesadores.

Muestran una evolución muy interesante, y a pesar de todo, no soy mucho de Intel, en la guerra moderna entre los procesadores que existen, yo soy del team AMD, me gustan más, aparte que ya superaron a Intel en todo.

Pero bueno, esta actividad muestra los avances que se hacían año con año, que demostraba la observación de Moore, que los procesadores incrementaban exponencialmente la cantidad de transistores por cada dos años y que por consiguiente aumentaban consecuentemente su potencia.

Una de las cosas que sí me di cuenta leyendo para esta investigación, fue que el procesador Celeron es muy malo, malísimo y si, lo confirmo, en otra pc de escritorio que tengo, lo tiene y si es muy malo.

Actividad 1: Parte II

Investigar los tipos de datos que emplean los microprocesadores Intel 80x86. Luego responde las siguientes preguntas sobre el conjunto de instrucciones para dichos microprocesadores, por cada pregunta incluye dos instrucciones de ejemplo explicando que función realiza

1- ¿Cuál es la función principal de las instrucciones aritméticas?

Suma ADD

Función Principal: La instrucción ADD se utiliza para sumar dos operandos. Estos operandos pueden ser registros, valores inmediatos o contenidos en una ubicación de memoria. El resultado de la suma se almacena en un registro específico o en una ubicación de memoria.

Unset

```
ADD R1, R2 ; Suma el contenido de R1 y R2, y almacena el resultado en R1.
```

Resta SUB

La instrucción SUB se utiliza para restar un operando de otro. Similar a ADD, los operandos pueden ser registros, valores inmediatos o posiciones de memoria. El resultado de la resta se almacena en un registro o memoria.

Unset

```
SUB R3, R4 ; Resta el contenido de R4 de R3, y almacena el resultado en R3.
```

Multiplicación MUL

La instrucción MUL realiza la multiplicación de dos operandos. A menudo, los operandos deben estar en registros específicos, y el resultado se almacena en un registro o conjunto de registros que pueden manejar el tamaño del resultado, ya que la multiplicación puede generar un valor mayor que el de los operandos originales.

Unset

```
MUL R5, R6 ; Multiplica el contenido de R5 por R6, y almacena el resultado en R5.
```

División DIV

La instrucción DIV realiza la división de un operando por otro. Como resultado, se obtienen el cociente y el residuo. El cociente se almacena en un registro específico, y el residuo (si es relevante) en otro registro.

Unset

```
DIV R7, R8 ; Divide el contenido de R7 entre R8, y almacena el cociente en R7 y el residuo en un registro asociado.
```

2.- ¿En qué consiste la conversión ASCII-BCD y en qué situación es útil?

Identifica el código ASCII del carácter que deseas convertir. Por ejemplo, si tienes el carácter '4', su representación ASCII es 0011 0100 (52 en decimal).

En la representación ASCII de un número (caracteres '0' a '9'), los cuatro bits menos significativos representan el valor del dígito en binario. Para '4' (0011 0100), los bits relevantes para el dígito son 0100, que es 4 en decimal.

Generación del BCD: Toma esos 4 bits y los utiliza como el dígito BCD correspondiente. En este caso, el BCD para '4' es simplemente 0100.

Otro ejemplo:

ASCII: '7' -> ASCII Code: 0011 0111

BCD: 0111

Utilidad:

Sistemas Embebidos y Dispositivos Electrónicos: Es útil en sistemas que requieren una conversión rápida y eficiente entre cadenas de texto y valores numéricos, especialmente en dispositivos que trabajan con representaciones binarias de números para la entrada o salida de datos.

En máquinas donde los números se deben procesar en su forma decimal, como en calculadoras y sistemas de contabilidad, la conversión a BCD es esencial para manejar y visualizar números de manera precisa.

En algunos sistemas más antiguos, la conversión entre diferentes representaciones numéricas como ASCII y BCD es necesaria para garantizar que los datos se transmiten y procesan correctamente entre diferentes dispositivos y programas.

3. Describe la función de las instrucciones de cambio de bit.

SET BIT: Función: Coloca un bit específico en 1 (encendido).

Unset
SETB bit

CLEAR BIT:

Función: Coloca un bit específico en 0 (apagado).

Unset
CLRB bit

TOGGLE BIT: Cambia el estado de un bit específico de 0 a 1 o de 1 a 0.

Unset
CPL bit

TEST BIT): Función: Verifica el valor de un bit específico, a menudo utilizado en combinación con instrucciones condicionales.

Unset
BTST bit

ROTATE BITS: Función: Desplaza los bits de un registro hacia la izquierda o derecha, con o sin acarreo.

Unset
ROL

4.¿Qué tipo de operaciones comparación se pueden realizar?

Igualdad: Verifica si dos valores son iguales.

Unset
CMP R1, R2 ; Compara el valor de R1 con R2.
; Si son iguales, se establece un indicador (flag) específico.

Desigualdad:

Verifica si dos valores no son iguales.

Unset
CMP R1, R2 ; Compara R1 con R2.
; Si no son iguales, se puede saltar a una instrucción
específica.

Mayor que:

Verifica si un valor es mayor que otro.

Unset
CMP R1, R2 ; Compara R1 con R2.
; Si R1 es mayor que R2, se establece un flag de mayor que.

Menor que:

Verifica si un valor es menor que otro.

Unset
CMP R1, R2 ; Compara R1 con R2.
; Si R1 es menor que R2, se establece un flag de menor que.

Mayor o igual que:

Verifica si un valor es mayor o igual a otro.

Unset

```
CMP R1, R2 ; Compara R1 con R2.  
           ; Si R1 es mayor o igual que R2, se establece el flag  
correspondiente.
```

A menudo, las comparaciones se combinan con instrucciones de salto condicional, como JZ (Jump if Zero), JNZ (Jump if Not Zero), JG (Jump if Greater), y JL (Jump if Less), para controlar el flujo de ejecución de un programa basándose en el resultado de la comparación.

5. Explica el propósito de las instrucciones de transferencia de datos.

En ensamblador y en general, las instrucciones de transferencia de datos buscan que se pueda mover la información para distintos fines, las más comunes son para procesar la información, la segunda es para almacenar la información.

6. ¿Qué función cumplen las instrucciones que operan sobre las banderas de dos estados?

Realmente lo que hacen es solo alterar el estado de dichas banderas, estas suelen indicar el resultado de comparaciones así como de resultados de operaciones aritméticas, esto suele ser lo normal, alterar el estado de las banderas puede ser beneficioso en casos muy particulares, pero si no se hace bien puede resultar terriblemente mal.

7. ¿Qué tipo de operaciones de entrada/salida se pueden realizar?

Se pueden manejar operaciones de entrada (Input) y de Salida (Output) y operaciones de Entrada y Salida (I/O)

8. ¿Cuál es el propósito de las instrucciones lógicas?

Las instrucciones lógicas en ensamblador tienen el propósito de realizar operaciones bit a bit en los datos almacenados en registros o en la memoria. Estas operaciones son fundamentales para manipular, evaluar y transformar datos a nivel de bits, lo que es esencial en muchos aspectos de la programación a bajo nivel, como la manipulación de datos, la toma de decisiones, y la optimización de operaciones.

9. ¿Cómo se utilizan las instrucciones de bucle?

Las instrucciones de bucle en ensamblador son fundamentales para la repetición de bloques de código, permitiendo realizar una operación varias veces de manera eficiente. En ensamblador, el bucle se controla mediante registros específicos que cuentan las iteraciones y las instrucciones de salto que determinan si el bucle debe continuar o terminar.

10. Explica la pila de datos y cómo se utilizan las instrucciones de pila.

La pila se usa para almacenar temporalmente valores que se necesitan más adelante, como en las operaciones aritméticas complejas o al intercambiar valores entre registros.

PUSH: Guarda un valor en la pila.

POP: Restaura un valor desde la pila.

CALL: Guarda la dirección de retorno y salta a una subrutina.

RET: Recupera la dirección de retorno y regresa de la subrutina.

PUSHF/POPF: Guarda/restaura el registro de banderas en/desde la pila.

11.¿Qué tipo de operaciones de cadena se pueden realizar?

En ensamblador, las operaciones de cadena permiten la manipulación eficiente de secuencias de datos (como cadenas de caracteres o bloques de memoria). Estas operaciones son útiles para copiar, comparar, mover, y procesar cadenas de datos con un número mínimo de instrucciones.

Unset

MOVS ; Mueve datos desde [SI/ESI/RSI] a [DI/EDI/RDI].

CMPSW ; Compara dos palabras de datos.

SCASB ; Compara AL con cada byte en la cadena.

LODSB ; Carga un byte en AL desde [SI/ESI/RSI].

STOSB ; Almacena el byte en AL en [DI/EDI/RDI].

REP MOVSB ; Mueve un bloque de bytes de una ubicación a otra, repitiendo CX veces.

12.Describe la dirección entre las instrucciones de transferencia condicional e incondicional.

Las instrucciones de transferencia de control en ensamblador permiten cambiar el flujo de ejecución de un programa.

Unset

JMP destino ; Salta a la etiqueta 'destino'

CALL subrutina ; Llama a la subrutina 'subrutina'

RET ; Retorna a la dirección guardada en la pila

JE destino ; Salta a 'destino' si ZF está activa

JNE destino ; Salta a 'destino' si ZF está desactivada

JG destino ; Salta a 'destino' si el primer operando es mayor

JL destino ; Salta a 'destino' si el primer operando es menor

13.¿Qué función cumplen las instrucciones de conversión de tipo?

Las instrucciones de conversión de tipo en ensamblador son esenciales para cambiar el formato o la representación de los datos en un programa. Estas conversiones son necesarias cuando se necesita transformar un dato de un tipo a otro, cómo cambiar un valor de un byte a una palabra, de un entero con signo a un entero sin signo, o de un número entero a uno en coma flotante.

Bibliografía:

x86 and amd64 instruction reference. (s. f.).

<https://www.felixcloutier.com/x86/>

Guide to x86 Assembly. (s. f.).

<https://flint.cs.yale.edu/cs421/papers/x86-asm/asm.html>

"Intel 64 and IA-32 Architectures Software Developer's Manual" - Intel.

Actividad 1: Parte III

Operaciones con cambio de base.

Ejercicios de Práctica: Conversión

Convierte los siguientes números decimales enteros a binario utilizando el formato de 16 Bits.

1. 256

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Posición
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	Binario
16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1	0	Decimal
Operación																
1.- 256 Decimal a Binario																
0000001000000000																

2. -128

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Posición
0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	Binario
16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1	0	Decimal
Operación																
1.- -128 Decimal a Binario																
0000000111111111																

3. 512

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Posición
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	Binario
16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1	0	Decimal
Operación																
3.- 512 a Binario																
0000010000000000																

4. -256

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Posición
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	Binario
16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1	0	Decimal
Operación																
4.--256 a Binario																
11111011111111																

5. 1023

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Posición
0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	Binario
16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1	0	Decimal
Operación																
5.-1023 a Binario																
0000011111111110																

Instrucciones: Realiza las siguientes operaciones de suma, resta y multiplicación, convirtiendo los números reales en números binarios de 16 bits y realizando las operaciones indicadas en binario.

1.- 25 + 12

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Posición
0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	Binario
16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1	0	Decimal
Operación																
25+12 a Binario																
0000000000110010																
0000000000011000																

0000000001001110																

2.- 38 + 56

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Posición
0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	Binario
16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1	0	Decimal
Operación																
38+56 a Binario																
0000000001001100																
0000000001110000																

0000000010111100																

3.- 7 * 9

```

      111
     1001
    -----
      111
     000
    000
   1111
  -----
0000000001111111

```

4.- $15+8$

```

      1111
       10
    -----
000000000010001

```

5.- $42 - (-20)$

```

      101010
      10100
    -----
0000000000111110

```

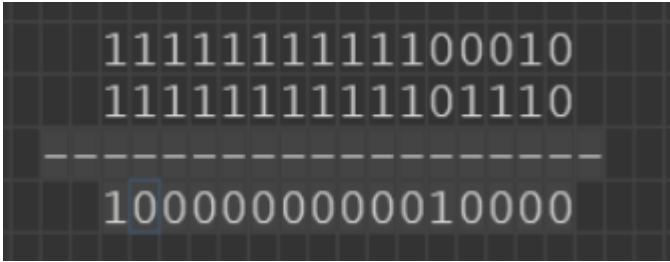
6.- $5 * 6$

```

      101
      110
    -----
      000
      101
      101
    -----
000000000011110

```

7.- $-30 + (-18)$



1111111111100010
1111111111101110

1000000000010000

The image shows a 20x5 grid of dark gray squares. The first row contains the binary string '1111111111100010'. The second row contains the binary string '1111111111101110'. The third row contains a dashed line '-----'. The fourth row contains the binary string '1000000000010000'. The first '1' in the fourth row is highlighted with a blue square.