• Otra forma de tratar es asignar prioridad a los distintas dispositivos de E/S, incluso el CPU debe asignarse su propia prioridad. De esta manera ⊛ un dispositivo de poca prioridad genera una interrupción puede ser ignorada mientras q' si un dispositivo de mayor prioridad es interrumpido este sera atendido de inmediato. Es por esta q' los rutinas de interrupciony para este caso, debe ser totalmente transparentes.

⊛ suponiendo q' ya hay una interrupción activa y.

16/09/25

Trabajo Practico N°2

1) $456789A7_{16}$

	10_{16}	11_{16}	12_{16}	13_{16}
Little endian	A1	89	67	45
Big endian	45	67	89	A1

$000058A_{16}$

	10_{16}	11_{16}	12_{16}	13_{16}
Little endian	8A	05	00	00
Big endian	00	00	05	8A

14148888_{16}

	10_{16}	11_{16}	12_{16}	13_{16}
Little endian	88	88	14	14
Big endian	14	14	88	88

2) Para determinar si el valor almacenado del entero es positivo o negativo tenemos q' ver dos cosas, primero el bit de signo y la forma de almacenamiento, es decir bigendian o littleendian

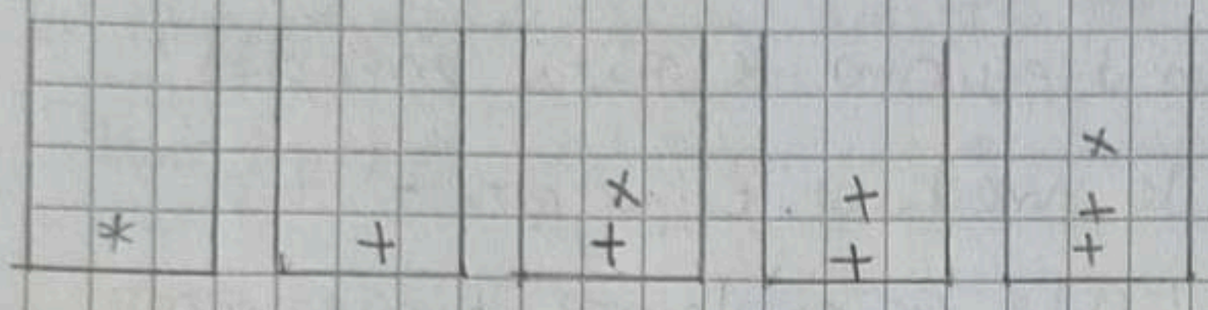
Si fuera <u>little endian</u> tendriamos primero ul par

$FB_{16} = (1111\ 1011)_2$, siendo el numero entonces negativo

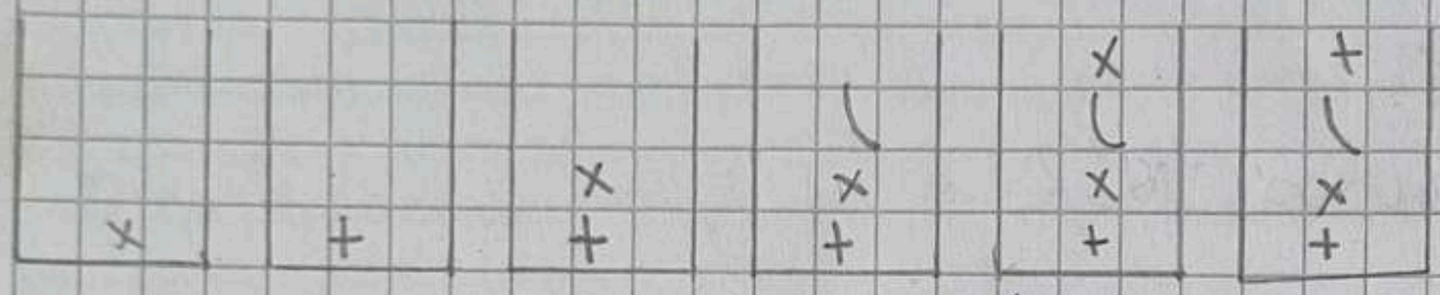
Si fuera Big endian tendriamos primero al par

$FZ_{16} = (1111\ 0010)_2$ → Quedaría el bit de signo 1 por lo tanto es negativo.

3) a) $X * Y + W * Z + V \times U =: \rightarrow$ $XY \times WZ \times VU \times + +$

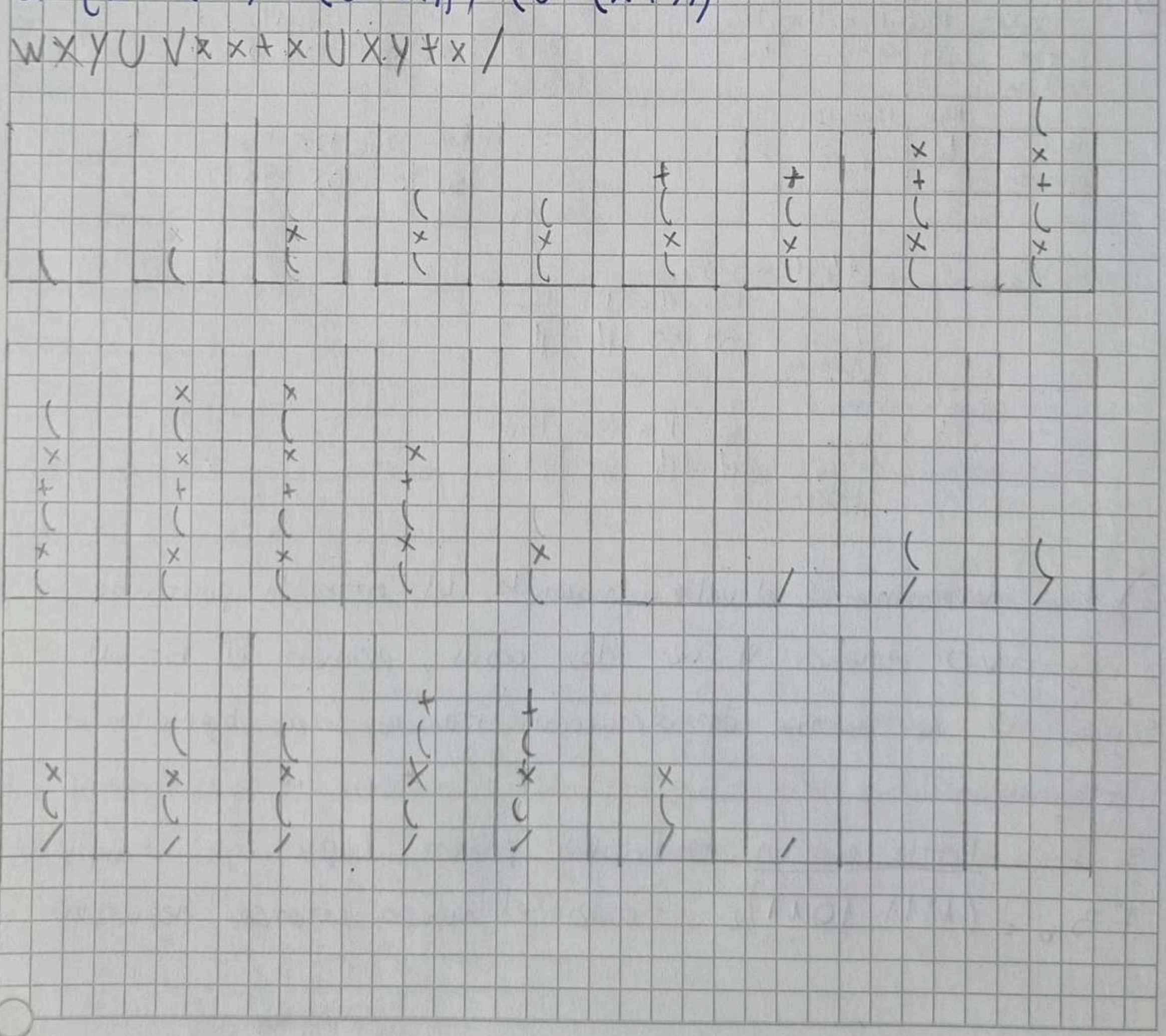


b) $W \times X + W \times (U \times V + Z) = \rightarrow$ $WX \times WUV \times + \times +$



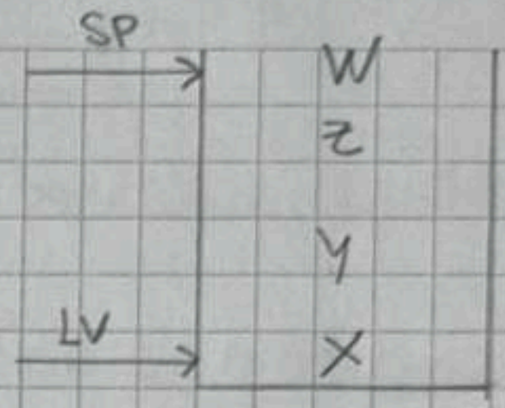
c) $(W \times (X + Y \times (U \times V))) / (U \times (X + Y)) =$

$WXYUV \times \times + \times UXY + \times /$



$x = x - (y+3) + z + (w-2)$

4) XY3+-Z+W2-+



ILOAD X	[x;]
ILOAD Y	[x; y]
BIPUSH 3	[x; y; 3]
IADD	[x; (y+3)]
ISUB	[x - (y+3)]
ILOAD Z	[x - (y+3); z]
IADD	[x - (y+3) + z]
ILOAD W	[x - (y+3) + z; w]
BIPUSH 2	[x - (y+3) + z; w; 2]
ISUB	[x - (y+3) + z; (w-2)]
IADD	[x - (y+3) + z + (w-2)]
ISTORE X	x = x - (y+3) + z + (w-2)

5) La principal diferencia entre el modo de direccionamiento directo y el modo indirecto es q el operando tiene la dirección de memoria efectiva para el modo directo, especial para declarar variables globales pues puede ser determinado en tiempo de compilación la ubicación en la memoria y siempre se mantendra en el mismo lugar en cada ejecución. Mientras q para el modo indirecto en el operando se tiene un registro el cual contiene la dirección efectiva de la memoria, este es un claro ejemplo de como funciona un puntero.

11 bit

b) | opcode | op1 | op2 |

3 bit 4 bit 4 bit

0 000
:
5 101 } 6 instrucciones de 2 direcciones

6 110 0000
:
110 1111
111 0000
:
35 111 1101 } 30 instrucciones de 1 dirección

36 111 1110 0000
:
111 1110 1111
111 1111 0000
:
59 111 1111 1000 } 24 instrucciones de 0 direcciones

8) modo inmediato: LOAD R3, 1400 R3 = 1400

Modo directo: LOAD R3, (1100) R3 = 400

Modo Indirecto: LOAD R3, (R2) R3 = 400

Modo Indexado: LOAD R3, (R1 + R2) R3 = 1100

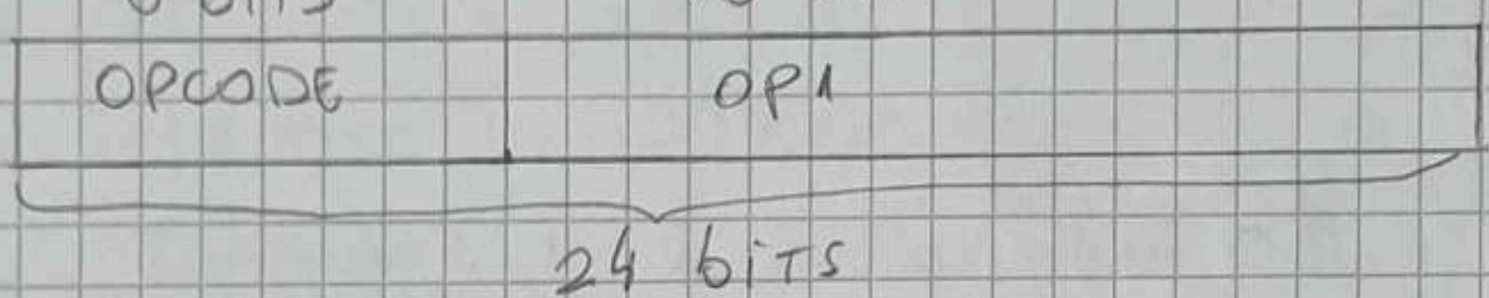
Modo Desplazamiento: LOAD 1000(R1) R3 = 1000

a) 1) Palabra: 24 bits

150 instrucciones de 1 dirección → OPCO DE = $\lceil \log_2 150 \rceil = 8$ bit

2) Para el operando quedaron: 24 bit - 8 bit = 16 bit

3) Dirección máxima de memoria = $2^{24} = 2^4 \cdot 2^{20}$ = 16 Mbits



10)

arquitectura MIPS (32 Release 5)

Modelo de Memoria

MIPS utiliza un espacio de direcciones de memoria plano y lineal de 32 bits. Es una arquitectura direccionable por bytes, lo que significa que cada byte tiene su propia dirección única. La arquitectura MIPS es por defecto **big-endian**, donde el byte más significativo se almacena en la dirección de memoria más baja.

Tipos de Datos

La arquitectura MIPS soporta los siguientes tipos de datos:

- **Enteros:** Bytes (8 bits), medias palabras (16 bits), palabras (32 bits) y dobles palabras (64 bits).
- **Punto Flotante:** Precisión simple (32 bits) y doble precisión (64 bits).

Modos de Direccionamiento

MIPS utiliza un conjunto reducido de modos de direccionamiento para simplificar el diseño de la CPU:

- **Inmediato:** El operando es una constante de 16 bits que está dentro de la misma instrucción.
- **Registro:** El operando es el contenido de uno de los 32 registros de propósito general.
- **Base (o Desplazamiento):** La dirección de memoria se calcula sumando una constante de 16 bits (desplazamiento) al valor de un registro base. Es el modo de direccionamiento más común para acceder a la memoria (ej. lw $t0, 8($s1)).
- **PC-relativo:** La dirección de destino se calcula sumando un desplazamiento al valor del contador de programa (PC). Se usa para ramificaciones (ej. beq).
- **Pseudodirecto:** La dirección de destino se forma combinando los bits más significativos del PC con un campo de dirección de 26 bits de la instrucción. Se usa para saltos incondicionales (ej. j).

Tipos de Operaciones y Control de Flujo

MIPS, al ser una arquitectura RISC, tiene un conjunto de instrucciones simplificado. Las operaciones se dividen en varias categorías:

- **Aritmético/Lógicas:** Operaciones como suma (add), resta (sub), AND (and), OR (or), etc.

- **Transferencia de Datos:** Solo las instrucciones load y store acceden a la memoria (ej. lw, sw).
- **Control de Flujo:** Incluye ramificaciones condicionales (ej. beq, bne), saltos incondicionales (j), y saltos a subrutinas (jal).

Formato de Instrucción

Las instrucciones MIPS tienen una longitud fija de 32 bits y se dividen en tres formatos principales para simplificar su decodificación:

- **Tipo R (Registro):** Usado para operaciones aritméticas y lógicas que solo involucran registros.
    - **Campos:** opcode (6 bits), rs (registro fuente 1), rt (registro fuente 2), rd (registro destino), shamt (bits de desplazamiento), y funct (código de función).
- **Tipo I (Inmediato):** Usado para operaciones con valores constantes, transferencias de datos y ramificaciones.
    - **Campos:** opcode (6 bits), rs (registro fuente), rt (registro destino/fuente), e immediate (valor de 16 bits).
- **Tipo J (Salto):** Usado para saltos incondicionales.
    - **Campos:** opcode (6 bits) y address (dirección de 26 bits).