UNIVERSIDAD NACIONAL DE JUJUY
FACULTAD DE INGENIERÍA

CÁTEDRA DE LABORATORIO DE COMPUTADORAS

LENGUAJE ASSEMBLER
(SET DE INSTRUCCIONES)

CARRERA DE INGENIERÍA INFORMÁTICA

LENGUAJE ASSEMBLER

JUEGO DE INSTRUCCIONES

1- ***Instrucciones de transferencia de datos:*** Mueven información entre registros y posiciones de memoria o puertos de entrada/salida.

INSTRUC	DESCRIPCIÓN	FORMATO	EJEMPLO
IN	Entrada de byte o palabra	**IN** acumulador, puerta	**IN** AX, 12h ;leer sobre AX una palabra de la puerta 12 h **IN** AL, DX ;leer sobre AL un byte de la puerta especificada en DX
LAHF	Carga el AH con las banderas	**LAHF**	; Carga los ocho bits de más a la derecha del registro de banderas y los envía al AH
LEA	Cargar dirección efectiva	**LEA** destino, fuente	**LEA** AX, XX[SI] ; si XX=1234h y SI = 0006h, AX=123Ah
LDS, LES, LFS, LGS, LSS	Carga el Registro de Segmento	**LDS/LES/LFS/LGS/LSS** registro, memoria	**LDS** puntero, AX ; se cargan al registro DS y al puntero una dirección de memoria especificada por AX
MOV	Mover	**MOV** destino, fuente	; AX = FEDCh - BX = 1234h **MOV** AX, BX ; AX = 1234h - BX = 1234h
OUT	Salida de byte o palabra	**OUT** puerta, acumulador	**OUT** 12h, AX ; transferir el valor de AX al puerto 12h **OUT** DX, AL ; transferir el valor de AL al puerto especificado en DX
POP	Recuperar palabra de la pila	**POP** destino	**POP** AX ; elemento de la pila a AX
POPF	Recuperar banderas de la pila	**POPF**	Transfiere bits específicos de la palabra que se encuentra en lo alto de la pila (apuntado por el registro SP) a las banderas, reemplazando así los valores que contenían previamente. El registro SP se incrementa luego en 2
PUSH	Depositar palabra en la pila	**PUSH** fuente	**PUSH** AX ; poner AX en la pila
PUSHF	Depositar banderas en la pila	**PUSHF**	Decrementa el puntero de pila (SP) en 2 y luego transfiere los valores de las banderas a bits específicos de la palabra de la pila direccionada por el registro SP
SAHF	Almacenar AH en banderas	**SAHF**	Transfiere bits específicos del registro AH a los registros de banderas SF, ZF, AF, PF y CF

INSTRUC	DESCRIPCIÓN	FORMATO	EJEMPLO
XCHG	Intercambiar dos operandos	**XCHG** destino, fuente	**XCHG AX, BX** ; AX = 1234h - BX = 5678h ; AX = 5678h - BX = 1234h
XLAT	Traslada bytes de una tabla a AL	**XLAT (mem)$_8$, XLATB**	TABLA DB 5,56,14,44,74,84 ;TABLA MOV BX, despl. TABLA ; Dsplazamiento de TABLA MOV AL, 2 ;AL apunta al elemento 3 XLAT TABLA ;AL=0EH

2- ***Instrucciones aritméticas:*** Realizan operaciones aritméticas sobre números binarios o números BCD (decimal codificado en binario)

INSTRUC	DESCRIPCIÓN	FORMATO	EJEMPLO
AAA	Ajuste ASCII en suma. *Corrige el resultado en AL de una suma de dos números decimales desempaquetados , convirtiéndolo en un valor decimal desempaquetado*	**AAA**	; AL = 08h AH = 00h ; BL = 06h **ADD** AL, BL ; AL = 0Eh AF = 0 ; BL = 06h **AAA** ; AX = 0104h AF = 1 CF = 1 ; BL = 06h
AAD	Ajuste ASCII en división *Realiza un ajuste del dividendo en AL antes de hacer la división de dos números decimales desempaquetados, para que el resultado de esta división (cociente) sea un valor decimal desempaquetado.*	**AAD**	; AX = 0103h = 259_{10} ; BL = 04h = 4_{10} **AAD** ; AX = 000Dh **DIV** BL ; AH = 01h (resto) ; AL = 03h (cociente)
AAM	Ajuste ASCII en multiplicación *Corrige el resultado en AX del producto de dos números decimales desempaquetados, convirtiéndolo en un valor decimal desempaquetado.*	**AAM**	; AL = 09h = 9_{10} ; BL = 07h = 7_{10} **MUL** BL ; AX = 003Fh = 63_{10} ; BL = 07h **AAM** ; AX = 0603h ; BL = 07h
AAS	Ajuste ASCII en resta *Corrige el resultado en AL de la resta de dos números decimales desempaquetados, convirtiéndolo en un valor decimal desempaquetado.*	**AAS**	; AL = 09h AH = 00h ; BL = 06h **SUB** AL,BL ; AL = 03h AF = 0 ; BL = 06h **AAS** ; AX = 0003h AF = 0 ; BL = 06h

INSTRUC	DESCRIPCIÓN	FORMATO	EJEMPLO
ADC	Sumar con acarreo *Suma los dos operandos. Suma uno si está activada la bandera de acarreo (CF=1). El resultado se almacena en el operando destino. Los operandos deben ser del mismo tipo (byte o palabra).*	**ADC** destino, fuente	**ADC** AL, BL ; AL = FEh = 1111 1110b ; BL = 12h = 0001 0010b ;---------- ; AL = 10h = 0001 0000b CF=1 ; CF = 1b ;---------- ; AL = 11h = 0001 0001b ; AL = 11h ; BL = 12h
ADD	Sumar sin acarreo *Suma los dos operandos. El resultado se almacena en el operando destino. Los operandos deben ser del mismo tipo (byte o palabra).*	**ADD** destino, fuente	**ADD** AL, BL ; AL = FEh = 1111 1110b ; BL = 12h = 0001 0010b ;---------- ; AL = 10h = 0001 0000b ; AL = 10h ; BL = 12h
CBW	Convierte byte a palabra. Extiende un número de un byte con signo a una palabra duplicando el bit de signo (bit 7) del AL a los bits en el AH.	**CBW**	**MOV** AL, 86h **CBW** ;AX = FF86h
CMP	Compara operandos *Resta fuente de destino, pero no retorna el resultado. Los operandos quedan inalterados, pero las banderas se actualizan, pudiéndose consultar a continuación mediante una instrucción de bifurcación condicional. Los operandos deben ser del mismo tipo (byte o palabra).*	**CMP** destino, fuente	**CMP** AX, BX ; comparar AX con BX **JL** MENOR ; bifurcar a MENOR si AX < BX
CWD	Convierte palabra en palabra doble. Extiende un número de una palabra con signo a una palabra doble en DX:AX duplicando el bit de signo (bit 15) del AX por medio del DX, por lo regular para generar un dividendo de 32 bits.	**CWD**	**MOV** AX, C234h **CWD** ; DX = FFFFh
DAA	Ajuste decimal en suma *Corrige el resultado en AL de la suma de dos números decimales empaquetados, convirtiéndolo en un valor decimal empaquetado.* *Si bits 3 a 0 de AL > 9 o AF = 1, AL = AL + 6, AF = 1;* *Si AL > 9Fh o CF = 1, AL = AL + 60h, CF = 1.*	**DAA**	**ADD** AL, BL ; AL = 68h ; BL = 17h ; AL = 7Fh AH = 00h AF = 0 **DAA** ; AX = 85h ; BL = 17h AF = 1

INSTRUC	DESCRIPCIÓN	FORMATO	EJEMPLO
DAS	Ajuste decimal en resta *Corrige el resultado en AL de la resta de dos números decimales empaquetados, convirtiéndolo en un valor decimal empaquetado.* *Si bits 3 a 0 de AL > 9 o AF = 1, AL = AL - 6, AF = 1;* *Si AL > 9Fh o CF = 1, AL = AL - 60h, CF = 1.*	**DAS**	; AL = 68h AH = 00h ; BL = 19h **SUB** AL, BL ; AL = 4Fh AF = 0 ; BL = 19h **DAS** ; AX = 49h AF = 1 ; BL = 19h
DEC	Decrementar destino en uno	**DEC** destino	; AX = 0123h **DEC** AX ; AX = AX – 1 = 0122h
DIV	Dividir sin signo. *Divide, sin considerar el signo, un número contenido en el acumulador y su extensión entre el operando fuente.* *El cociente se almacena en el acumulador (AL o AX según que el operando fuente sea de tipo byte o palabra).* *El resto se almacena en la extensión del acumulador (AH o DX, según que el operando fuente sea de tipo byte o palabra).*	**DIV** fuente	; AX = 0013h = 19_{10} (dividendo) ; BL = 02h = 2_{10} (divisor) **DIV** BL ; AX = 0109h (resultado) ; AH = 01h = 1_{10} (resto) ; AL = 09h = 9_{10} (cociente) ; BL = 02h
IDIV	Dividir con signo *Divide, considerando el signo, un número contenido en el acumulador y su extensión entre el operando fuente.* *El cociente se almacena en el acumulador (AL o AX según que el operando fuente sea de tipo byte o palabra).* *El resto se almacena en la extensión del acumulador (AH o DX, según que el operando fuente sea de tipo byte o palabra).*	**IDIV** fuente	; AX = FFEDh = -19_{10} (dividendo) ; BL = 02h = 2_{10} (divisor) **IDIV** BL ; AX = 01F7h (resultado) ; AH = 01h = 1_{10} (resto) ; AL = F7h = -9_{10} (cociente) ; BL = 02h
IMUL	Multiplicar con signo *Multiplica, considerarando el signo, el acumulador (AL o AX) por el operando fuente. Si la mitad superior del resultado (AH para el caso de operando tipo byte o DX para el caso de operando de tipo palabra) no es la expansión del signo del resultado, se activan las banderas CF y OF, indicando que esta mitad superior contiene dígitos significativos del resutlado.*	**IMUL** fuente	; AL = FEh = -2_{10} ; BL = 12h = 18_{10} **IMUL** BL ; AX = FFDCh (resultado) ; BL = 12h ; CF = OF = 0
INC	Incrementar destino en uno	**INC** destino	; AX = 1234h **INC** AX ; AX = AX + 1 = 1235h
MUL	Multiplicar sin signo *Multiplica, sin considerar el signo, el acumulador (AL o AX) por el operando fuente. Si la mitad superior del resultado (AH para el caso de operando tipo byte o DX para el caso de operando de tipo palabra) no es cero, se activan las banderas CF y OF, indicando que esta mitad superior contiene dígitos significativos del resultado.*	**MUL** fuente	; AL = 64h = 100_{10} ; BL = 02h = 2_{10} ; 00C8h = 200_{10} **MUL** BL ; AX = 00C8h (resultado) ; BL = 02h ; CF = OF = 0

INSTRUC	DESCRIPCIÓN	FORMATO	EJEMPLO
NEG	Negar/formar complemento a 2. Calcula el valor negativo del operando destino, es decir, resta el operando de cero y devuelve el resultado en el mismo operando (byte o palabra). Para hacer esto, el operando destino se resta del número compuesto por todos unos y se le añade 1. Esto es lo mismo que el complemento a 2 del número.	**NEG** destino	**NEG** AL ; FFh = 1111 1111b ; AL = F2h = 1111 0010b ; 0Dh = 0000 1101b ; +1b ; AL = 0Eh = 0000 1110b
SBB	Restar con acarreo *Resta el operando fuente del operando destino.* *Resta uno si está activada la bandera de acarreo (CF). El resultado se almacena en el operando destino.* *Los operandos deben ser del mismo tipo (byte o palabra)*	**SBB** destino, fuente	**SBB** AL, BL ; AL = FEh = 1111 1110b ; BL = 12h = 0001 0010b ; ECh = 1110 1100b ; CF = 01h = - 1b ; EBh = 1110 1011b ; AL = EBh ; BL = 12h
SUB	Restar sin acarreo *Resta el operando fuente del operando destino.* *El resultado se almacena en el operando destino.* *Los operandos deben ser del mismo tipo (byte o palabra)*	**SUB** destino, fuente	**SUB** AL, BL ; AL = FEh = 1111 1110b ; BL = 12h = 0001 0010b ; ECh = 1110 1100b ; AL = ECh ; BL = 12h

3- ***Instrucciones de manejo de bits:*** Realizan operaciones de desplazamiento, rotación y lógicas sobre registros o posiciones de memoria.

INSTRUC	DESCRIPCIÓN	FORMATO	EJEMPLO
AND	Y lógico Operación "y lógica" a nivel de bit entre los dos operandos. El resultado se almacena en destino.	**AND** destino, fuente	**AND** AX, BX ; AX = FEDCh = 1111 1110 1101 1100b ; BX = 1234h = 0001 0010 0011 0100b ; 1214h = 0001 0010 0001 0100b ; AX = 1214h ; BX = 1234h
NOT	NO lógico *Cambia los bits unos por ceros y los bits ceros por unos, es decir, realiza el complemento a unos del operando y devuelve el resultado en el mismo operando (byte o palabra)*	**NOT** destino	**NOT** AL ; AL = F2h = 1111 0010b ; AL = 0Dh = 0000 1101b
OR	O lógico Operación "o lógico inclusivo" a nivel de bit entre los dos operandos. El resultado se almacena en destino.	**OR** destino, fuente	**OR** AX, BX ; AX = FEDCh = 1111 1110 1101 1100b ; BX = 1234h = 0001 0010 0011 0100b ; FEFCh = 1111 1110 1111 1100b ; AX = FEFCh ; BX = 1234h

INSTRUC	DESCRIPCIÓN	FORMATO	EJEMPLO
RCL	Rotar a la izquierda a través de acarreo *Rotar a la izquierda los bits del operando destino junto con la bandera de acarreo (CF) el número de bits especificado en el segundo operando.* *Si el número de bits a desplazar es 1, se puede especificar directamente. Si es mayor que 1, su valor debe cargarse en CL y especificar CL como segundo operando.*	**RCL** destino, contador	**MOV** CL, 3 ; rotar 3 bits ; AL = 0101 1110b, CF = 0 **RCL** AL, CL ; AL = 1111 0001b, CF = 0
RCR	Rotar a la derecha a través de acarreo *Rotar a la derecha los bits del operando destino junto con la bandera de acarreo (CF) el número de bits especificado en el segundo operando.* *Si el número de bits a desplazar es 1, se puede especificar directamente. Si es mayor que 1, su valor debe cargarse en CL y especificar CL como segundo operando.*	**RCR** destino, contador	**MOV** CL, 3 ; rotar 3 bits ; AL = 0101 1110b, CF = 0 **RCL** AL, CL ; AL = 1000 1011b, CF = 1
ROL	Rotar a la izquierda *Rotar a la izquierda los bits del operando destino el número de bits especificado en el segundo operando.* *Si el número de bits a desplazar es 1, se puede especificar directamente. Si es mayor que 1, su valor debe cargarse en CL y especificar CL como segundo operando.*	**ROL** destino, contador	**MOV** CL, 2 ; rotar 2 bits ; AL = 1100 1100b, CF = 0 **RCL** AL, CL ; AL = 0011 0011b, CF = 1
ROR	Rotar a la derecha *Rotar a la derecha los bits del operando destino el número de bits especificado en el segundo operando.* *Si el número de bits a desplazar es 1, se puede especificar directamente. Si es mayor que 1, su valor debe cargarse en CL y especificar CL como segundo operando.*	**ROR** destino,contador	**MOV** CL, 2 ; rotar 2 bits ; AL = 1100 1100b, CF = 0 **RCR** AL, CL ; AL = 0011 0011b, CF = 0
SAL	Desplazamiento aritmético a la izquierda	**SAL** destino, contador **SHL** destino, contador	**MOV** CL, 2 ; desplazar 2 bits ; AL = 1100 1100b, CF = 0 **SAL** AL, CL ; AL = 0011 0000b, CF = 1
SHL	* Desplazamiento lógico a la izquierda (igua a SAL) *Desplazan a la izquierda los bits del operando destino el número de bits especificado en el segundo operando.* *Los bits de la derecha se rellenan con cero*		
SAR	Desplazamiento aritmético a la derecha *Desplaza a la derecha los bits del operando destino el número de bits especificado en el segundo operando.* *Los bits de la isquierda se rellenan con el bit de signo del primer operando.*	**SAR** destino, contador	**MOV** CL, 2 ; desplazar 2 bits ; AL = 1100 1100b, CF = 0 **SAR** AL, CL ; AL = 1111 0011b, CF = 0
SHR	Desplazamiento lógico a la derecha *Desplazar a la derecha los bits del operando destino el número de bits especificado en el segundo operando.* *Los bits de la izquierda se rellenan con cero.*	**SHR** destino, contador	**MOV** CL, 2 ; desplazar 2 bits ; AL = 0011 0011b, CF = 0 **SHR** AL, CL ; AL = 0000 1100b, CF = 1
TEST	Comparación lógica. Efectúa la operación lógica AND, bit a bit, entre los dos operandos de que dispone. El resultado se pierde y sólo quedan afectados los señalizadores SF, ZF y PF, mientras que OF=CF=0 quedando indefinido AF.	**TEST** {reg/mem},{reg/inm/mem}	**TEST AX, FFFFH** ; Comparación lógica, bit por bit, entre el contenido del AX y el valor FFFFh.

INSTRUC	DESCRIPCIÓN	FORMATO	EJEMPLO
XOR	O lógico exclusivo *Operación "o lógico exclusivo" a nivel de bit entre los dos operandos. El resultado se almacena en destino.*	**XOR** destino, fuente	; AX = FEDCh = 1111 1110 1101 1100b ; BX = 1234h = 0001 0010 0011 0100b ; ECD8h = 1110 1100 1110 1000b **XOR** AX, BX ; AX = ECD8h ; BX = 1234h

4- ***Instrucciones de transferencia de control:*** Sirven para controlar la secuencia de ejecución de las instrucciones del programa.

INSTRUC	DESCRIPCIÓN	FORMATO	EJEMPLO
CALL	Llamar a un procedimiento *Bifurca a un procedimiento fuera de línea, salvando previamente en la pila la dirección de la instrucción siguiente, para poder volver a esta instrucción una vez ejecutado el procedimiento*	**CALL** destino	**CALL** Proc1 ; llamada directa al procedimiento Proc1
JA	Bifurcar si superior	**JA** desplazamiento **JNBE** desplazamiento	**CMP** AX, BX ; comparar AX con BX **JA** ETIQUETA ; bifurcar a ETIQUETA si AX > BX ; (sin considerar el signo)
JNBE	* Bifurcar si no inferior ni igual (igual a JA) *Transfiere el control a la instrucción (IP + desplazamiento) si se cumple la condición CF = 0 y ZF = 0.* *Si CF = 1 o ZF = 1, no hay transferencia de control.*		
JAE	Bifurcar si superior o igual	**JAE** desplazamiento **JNB** desplazamiento	**CMP** AX, BX ; comparar AX con BX **JAE** ETIQUETA ; bifurcar a ETIQUETA si AX >= BX ; (sin considerar el signo)
JNB	* Bifurcar si no inferior (igual a JAE) *Transfiere el control a la instrucción (IP + desplazamiento) si se cumple la condición CF = 0 .* *Si CF = 1, no hay transferencia de control.*		
JB	Bifurcar si inferior	**JB** desplazamiento **JNAE** desplazamiento **JC** desplazamiento	**CMP** AX, BX ; comparar AX con BX **JB** ETIQUETA ; bifurcar a ETIQUETA si AX < BX ; (sin considerar el signo)
JNAE	* Bifurcar si no superior ni igual (igual a JB)		
JC	* Bifurcar si acarreo (igual a JB) *Transfiere el control a la instrucción (IP + desplazamiento) si se cumple la condición CF = 1 .* *Si CF = 0, no hay transferencia de control.*		
JBE	Bifurcar si inferior o igual	**JBE** desplazamiento **JNA** desplazamiento	**CMP** AX, BX ; comparar AX con BX **JBE** ETIQUETA ; bifurcar a ETIQUETA si AX <= BX ; (sin considerar el signo)
JNA	* Bifurcar si no superior (igual a JBE) *Transfiere el control a la instrucción (IP + desplazamiento) si se cumple la condición CF = 1 o ZF = 1.* *Si CF = 0 y ZF = 0, no hay transferencia de control.*		
JCXZ	Bifurcar si CX es cero *Transfiere el control a la instrucción (IP + desplazamiento) si se cumple la condición CX = 0.* *Si CF <> 0, no hay transferencia de control*	**JCXZ** desplazamiento	**DEC** CX ; CX = CX – 1 **JCXZ** ETIQUETA ; bifurcar a ETIQUETA si CX = 0

INSTRUC	DESCRIPCIÓN	FORMATO	EJEMPLO
JE	Bifurcar si igual	**JE** desplazamiento **JZ** desplazamiento	**CMP** AX, BX ; comparar AX con BX **JE** ETIQUETA1 ; bifurcar a ETIQUETA1 si AX= BX **SUB** AX, BX ; AX = AX – BX **JZ** ETIQUETA2 ; bifurcar a ETIQUETA2 si AX es 0
JZ	* Bifurcar si igual a cero (igual a JE) *Transfiere el control a la instrucción (IP + desplazamiento) si se cumple la condición ZF = 1.* *Si ZF = 0, no hay transferencia de control*		
JG	Bifurcar si mayor	**JG** desplazamiento **JNLE** desplazamiento	**CMP** AX, BX ; comparar AX con BX **JG** ETIQUETA ; bifurcar a ETIQUETA si AX >BX
JNLE	* Bifurcar si no menor ni igual (igual a JG) *Transfiere el control a la instrucción (IP + desplazamiento) si se cumple la condición ZF = 0 y SF = OF.* *Si ZF = 1 o SF<>OF, no hay transferencia de control*		
JGE	Bifurcar si mayor o igual	**JGE** desplazamiento **JNL** desplazamiento	**CMP** AX, BX ; comparar AX con BX **JGE** ETIQUETA ; bifurcar a ETIQUETA si AX >=BX ; (considerando el signo)
JNL	* Bifurcar si no menor (igual a JGE) *Transfiere el control a la instrucción (IP + desplazamiento) si se cumple la condición SF = OF.* *Si SF<>OF, no hay transferencia de control.*		
JL	Bifurcar si menor	**JL** desplazamiento **JNGE** desplazamiento	**CMP** AX, BX ; comparar AX con BX **JL** ETIQUETA ; bifurcar a ETIQUETA si AX < BX ; (considerando el signo)
JNGE	* Bifurcar si no mayor ni igual (igual a JL) *Transfiere el control a la instrucción (IP + desplazamiento) si se cumple la condición SF <> OF.* *Si SF=OF, no hay transferencia de control.*		
JLE	Bifurcar si menor o igual	**JLE** desplazamiento **JNG** desplazamiento	**CMP** AX, BX ; comparar AX con BX **JLE** ETIQUETA ; bifurcar a ETIQUETA si AX <=BX ; (considerando el signo)
JNG	* Bifurcar si no mayor (igual a JLE) *Transfiere el control a la instrucción (IP + desplazamiento) si se cumple la condición ZF = 1 o SF<>OF.* *Si ZF = 0 y SF = OF, no hay transferencia de control.*		
JNC	* Bifurcar si no acarreo (igual a JAE) *Transfiere el control a la instrucción (IP + desplazamiento) si se cumple la condición CF = 0.* *Si CF = 1, no hay transferencia de control.*	**JNC** desplazamiento	**ADD** AL, BL ; AL = AL + BL **JNC** ETIQUETA ; bifurcar a ETIQUETA si no acarreo
JNE	Bifurcar si no igual	**JNE** desplazamiento **JNZ** desplazamiento	**CMP** AX, BX ; comparar AX con BX **JNE** ETIQUETA1 ; bifurcar a ETIQUETA1 si distintos **SUB** AX, BX ; AX = AX – BX **JNZ** ETIQUETA2 ; bifurcar si AX es distinto de 0
JNZ	* Bifurcar si no cero (igual a JNE) *Transfiere el control a la instrucción (IP + desplazamiento) si se cumple la condición ZF = 0.* *Si ZF = 1, no hay transferencia de control.*		

INSTRUC	DESCRIPCIÓN	FORMATO	EJEMPLO
JNO	Bifurcar si no desbordamiento (overflow) *Transfiere el control a la instrucción (IP + desplazamiento) si se cumple la condición OF = 0.* *Si OF = 1, no hay transferencia de control.*	**JNO** desplazamiento	**ADD** AL, BL ; AL = AL + BL **JNO** ETIQUETA ; bifurcar si no desbordamiento ; (overflow)
JNP	Bifurcar si no paridad	**JNP** desplazamiento **JPO** desplazamiento	**JNP** ETIQUETA ; bifurcar si paridad impar
JPO	* Bifurcar si paridad impar (igual a JNP) *Transfiere el control a la instrucción (IP + desplazamiento) si se cumple la condición PF = 0.* *Si PF = 1, no hay transferencia de control*		
JNS	Bifurcar si no signo/si positivo *Transfiere el control a la instrucción (IP + desplazamiento) si se cumple la condición SF = 0.* *Si SF = 1, no hay transferencia de control*	**JNS** desplazamiento	**SUB** AX, BX ; AX = AX – BX **JNS** ETIQUETA ; bifurcar a ETIQUETA si no signo, ; es decir si AX >=BX
JMP	Bifurcar incondicinalmente. *Transfiere el control incondicionalmente al operando.*	**JMP** desplazamiento	**JMP** ETIQUETA ; bifurcación directa a ETIQUETA
JO	Bifurcar si desbordamiento (overflow) *Transfiere el control a la instrucción (IP + desplazamiento) si se cumple la condición OF = 1.* *Si CF = 0, no hay transferencia de control.*	**JO** desplazamiento	**ADD** AX, BX ; AX = AX – BX **JO** ETIQUETA ; bifurcar a ETIQUETA si desbord. ; (overflow)
JP	Bifurcar si paridad	**JP** desplazamiento **JPE** desplazamiento	**JP** ETIQUETA ; bifurcar si paridad par
JPE	* Bifurcar si paridad par (igual a JP) *Transfiere el control a la instrucción (IP + desplazamiento) si se cumple la condición PF = 1.* *Si PF = 0, no hay transferencia de control.*		
JS	Bifurcar si signo *Transfiere el control a la instrucción (IP + desplazamiento) si se cumple la condición SF = 1.* *Si SF = 0, no hay transferencia de control.*	**JS** desplazamiento	**SUB** AX, BX ; AX = AX – BX **JS** ETIQUETA ; bifurcar a ETIQUETA si signo, es ; decir, si AX < 0
LOOP	Bucle hasta que se acabe el contador *Decrementa el registro contador (CX). Si CX es distinto de cero, entonces IP = IP + desplazamiento (expandiendo el signo a 16 bits). Si CX es cero, entonces se ejecuta la siguiente instrucción.*	**LOOP** desplazamiento	
LOOPE	Bucle mientras igual	**LOOPE** desplazamiento **LOOPZ** desplazamiento	
LOOPZ	* Bucle mientras cero (igual a LOOPE) *Decrementa el registro contador (CX)* *Si ZF = 1 y CX <> 0, entonces IP = IP + desplazamiento* *Si ZF = 0 o CX = 0, entonces se ejecuta la instrucción siguiente.*		

INSTRUC	DESCRIPCIÓN	FORMATO	EJEMPLO
LOOPNE	Bucle mientras no igual	**LOOPNE** desplazamiento **LOOPNZ** desplazamiento	
LOOPNZ	* Bucle mietnras no cero (igual a LOOPNE) *Decrementa el registro contador (CX)* *Si ZF = 0 y CX <> 0, entonces IP = IP + desplazamiento* *Si ZF = 1 o CX = 0, entonces se ejecuta la instrucción siguiente.*		
RET	Retornar de un procedimiento *Retorna de un procedimiento, previamente invocado mediante CALL, utilizando como dirección de retorno la dirección salvada en la pila por CALL, que corresponde a la instrucción siguiente a dicha sentencia CALL.*	**RET** [valor]	**RET** ; retorno de procedimiento ; (no hay parámetros)

5- ***Instrucciones de manejo de cadenas (strings):*** Realizan operaciones sobre series de bytes o palabras como mover, comparar y explorar.

INSTRUC	DESCRIPCIÓN	FORMATO	EJEMPLO
CMPS	Comparar cadenas de bytes o palabras	**CMPS** destino, fuente	**LEA** SI, FUENTE ; SI = desplazamiento de FUENTE **LEA** DI, DESTINO ; DI = desplazamiento de DESTINO **CMPS** [DI], [SI] ; comparar FUENTE con DESTINO **JL** PEPE bifurca si FUENTE < DESTINO
CMPSB	* Comparar cadenas de bytes (igual a CMPS)	**CMPSB** destino, fuente	
CMPSW	* Comparar cadenas de palabras (igual a CMPS)	**CMPSW** destino, fuente	
LODS	Cargar cadena de bytes o palabras	**LODS** cadena_fuente	**CLD** ; DF = 0 (incrementar SI) **LEA** SI, FUENTE ; SI = desplazamiento de FUENTE **LODS** FUENTE ; AL = FUENTE, o **LODSB** FUENTE ; AL = DS:[SI], SI = SI + 1, o **LODSW** FUENTE ; AL = DS:[SI], SI = SI + 2 ; según FUENTE sea byte o palabra
LODSB	* Cargar cadena de bytes (igual a LODS)	**LODSB** cadena_fuente	
LODSW	* Cargar cadena de palabras (igual a LODS)	**LODSW** cadena_fuente	
MOVS	Mover cadena de bytes o palabras	**MOVS** destino, fuente	**CLD** ; DF = 0 **LEA** SI, FUENTE ; SI = desplazamiento de FUENTE **LEA** DI, DESTINO ; DI = desplazamiento de DESTINO **MOV** CX, 100 ; CX = 100 (número de elementos) **REP MOVS** [DI], [SI] ; mover los elementos indistintos, o **REP MOVSB** [DI], [SI] ; mover los elementos de tipo byte, o **REP MOVSW** [DI],[SI] ; mover los de tipo palabra
MOVSB	* Mover cadena de bytes (igual a MOVS)	**MOVSB** destino, fuente	
MOVSW	* Mover cadena de palabras (igual a MOVS)	**MOVSW** destino,fuente	

INSTRUC	DESCRIPCIÓN	FORMATO	EJEMPLO
REP	Repetir operación de cadena	**REP**	**MOV** CX, 100 ; CX = longitud de las tablas **LEA** SI, TABLA1 ; SI = desplazamiento de TABLA1 ; (segmento de datos) **LEA** DI, TABLA2 ; DI = desplazamiento de TABLA2 ; (segmento extra) **REP MOVS** [DI], [SI] ; copiar elementos
REPE	Repetir operación de cadena, mientras igual	**REPE**	
REPNE	Repetir operación de cadena, mientras no igual	**REPNE**	
REPNZ	* Repetir operación de cadena, mientras no cero (igual a REPNE)	**REPNZ**	
REPZ	* Repetir operación de cadena, mientras cero (igual a REPE)	**REPZ**	
SCAS	Explorar cadena de bytes o palabras	**SCAS** cadena_destino	**CLD** ; DF = 0 (incrementar DI) **LEA** DI, DESTINO ; DI = desplazamiento de DESTINO **MOV** CX, 100 ; CX = 100 (número de elementos) **MOV** AL, 50 ; AL = 50 (valor a buscar) **REPE SCAS** DESTINO ; explorar cadena
SCASB	* Explorar cadena de bytes (igual a SCAS)	**SCASB** cadena_destino	
SCASW	* Explorar cadena de palabras (igual a SCAS)	**SCASW** cadena_destino	
STOS	Almacenar cadena de bytes o palabras	**STOS** cadena_destino	**CLD** ; DF = 0 (incrementar DI) **LEA** DI, DESTINO ; DI = desplazamiento de DESTINO **STOS** DESTINO ; DESTINO = AL o AX, o **STOSB** DESTINO ; AL = ES:[DI], DI = DI + 1, si byte **STOSW** DESTINO ; AX = ES:[DI], DI = DI + 2, si palab
STOSB	* Almacenar cadena de bytes (igual a STOS)	**STOSB** cadena_destino	
STOSW	* Almacenar cadena de palabras (igual a STOS)	**STOSW** cadena_destino	

6- ***Instrucciones de interrupción:*** Provocan la interrupción del microprocesador para que realice un servicio determinado.

INSTRUC	DESCRIPCIÓN	FORMATO	EJEMPLO
INT	Interrupción. Interrumpe el procesamiento y transfiere el control a uno de las 256 direcciones (vector) de interrupción empezando en el segmento 0, desplazamiento 0. realiza lo siguiente: (1) guarda el la pila las banderas y restablece las banderas IF y TF; (2) guarda el la pila el CS y coloca la palabra de orden alto de la dirección de interrupción en el CS; y (3) guarda en la pila el IP y llena el IP con la palabra de orden bajo de la dirección de interrupción.	**INT** tipo_interrupción	INT 23 ; interrupción 23
INTO	Interrumpe un desbordamiento. Provoca una interrupción (generalmente inofensiva) si ha ocurrido un desbordamiento (la OF está en 1) y realiza una INT 04h. La dirección de interrupción está en la localidad 10h de la tabla de servicio de interrupción.	**INTO**	; provoca una interrupción si el señalizador OF = 1
IRET **IRETD**	Retorno de interrupción. Realiza el siguiente procedimiento: (1) saca de la pila la palabra de la parte superior de la pila al IP, incrementa en 2 el SP y saca de la parte superior de la pila al CS; (2) incrementa en 2 el SP y saca de la pila al registro de bandera. Este proceso deshace los pasos de la interrupción original y realiza un regreso.	**IRET**	; retorno de interrupción

7- ***Instrucciones de control del microprocesador:*** Activan y desactivan banderas y cambian el estado de ejecución del microprocesador.

INSTRUC	DESCRIPCIÓN	FORMATO	EJEMPLO	
CLC	Borrar bandera de acarreo	**CLC**	**CLC**	; CF = 0
CLD	Borrar bandera de dirección *Poner a 0 la bandera de dirección (DF), por lo que en la ejecución de las instrucciones de manejo de cadenas los registros índices SI y/o DI se autoincrementan de modo automático*	**CLD**	**CLD**	; incrementar índices en ;instrucciones de manejo de cadenas
CLI	Borrar bandera de interrupción	**CLI**	**CLI**	; desactivar interrupciones ;enmascarables
CMC	Complementar bandera de acarreo *Pone la bandera de acarreo en el estado contrario.*	**CMC**	**CMC**	; complementar CF
ESC	Escape (transmitir información a un coprocesador)	**ESC** código, fuente	**WAIT** **ESC** 21h, TABLA	; esperar a que el coprocesador ; acabe con la operación en curso ; bus de datos = contenido de ; TABLA
HLT	Parar el procesador	**HLT**	**HLT**	; para el procesador
LOCK	Impide acceso al bus.	**LOCK** instrucción	**LOCK** SCHG AX, MEM	; ejecuta la instrucción completa sin posibilidad de ser interrumpida por una petición del bus de otro maestro.
NOP	No operación.	**NOP**	**NOP**	; no realiza ninguna operación
STC	Poner bandera de acarreo	**STC**	**STC**	; CF = 1
STD	Poner bandera de dirección *Poner a 1 la bandera de acarreo (DF), por lo que en la ejecución de las instrucciones de manejo de cadenas los registros ínidces SI y/o DI se autodecrementan de modo automático*	**STD**	**STD**	; decrementar índices en ;instrucciones de manejo de cadenas
STI	Poner bandera de permitir interrupciones	**STI**	**STI**	; activar interrupciones ; enmascarables
WAIT	Esperar que acabe un coprocesador	**WAIT**	**WAIT** **ESC** 21h, TABLA	; esperar a que el coprocesador esté ; disponible ;pasar información al coprocesador

DIRECTIVAS

Las directivas o pseudooperaciones se pueden dividir en cuatro grupos funcionales:

1.- Directivas de Datos

2.- Directivas Condicionales

3.- Directivas de Listado

4.- Directivas de macros (no incluidas)

1- ***DIRECTIVAS DE DATOS:*** A su vez se dividen en las categorías siguientes:

- ***Definición de Símbolos:*** sirven para asignar nombres simbólicos a expresiones. Una vez definido el símbolo se puede usar dicho símbolo en lugar de la expresión equivalente.

DIRECTIVA	DESCRIPCIÓN	FORMATO	EJEMPLO
EQU	**Asigna un símbolo a una expresión fija** *La utilidad de esta directiva reside en hacer más claras las sentencias fuentes ensamblador.*	nombre **EQU** expresión	COLUMNAS **EQU** 80 ; longitud de una línea FILAS **EQU** 25 ; número de filas PANTALLA **EQU** FILAS*COLUMNAS ; tamaño de la pantalla CR **EQU** 13 ; retorno de carro LF **EQU** 10 ; alimentación de línea CONTADOR **EQU** CX ; nombre alternativo de CX MOVER **EQU** MOV ; nombre alternativo de MOV
=	**Asigna un símbolo a una expresión fija** *Es similar a EQU, excepto que "nombre" puede redefinirse.*	nombre **=** expresión	VALOR = 10 ; nombre de la constante 10 VALOR = VALOR + 1 ; utiliza la definición anterior, ahora ; VALOR es 11

- ***Definición de Datos:*** sirven para reservar memoria para las variables del programa. Opcionalmente se puede dar un valor inicial a cada variable.

DIRECTIVA	DESCRIPCIÓN	FORMATO	EJEMPLO
DB	**Definir byte** *Reserva memoria para una variable tipo byte (8 bits)*	[variable] **DB** expresión [,...]	VALORES **DB** 30, -15, 20 MAXSS **DB** 255 ; número máximo sin signo **DB** 0Dh ; retorno de carro **DB** 12 * 3 ; expresión de constantes **DB** 4 DUP (0) ; es equivalente a DB 0,0,0,0 **DB** 4 DUP (3 DUP (4),7) ; es equivalente a DB 4 DUP (4,4,4,7)
DW	**Definir palabra** *Reserva memoria para una variable tipo palabra (16 bits)*	[variable] **DW** expresión [,...]	VALORES **DW** 300, -150, 2000 MAXSS **DW** 65535 ; numero máximo sin signo **DW** 120 * 3 ; expresión de constantes **DW** 4 DUP (0) ; es equivalente a DD 0,0,0,0 TABLA **DW** 4, ?, 650 ; no se define el segundo valor
DD	**Definir doble palabra** *Reserva memoria para una variable tipo doble palabra (32 bits)*	[variable] **DD** expresión [,...]	VALORES **DD** 300, -150, 2000 MAXSS **DD** 4294967295 ; numero máximo sin signo **DD** 120 * 3 ; expresión de constantes **DD** 4 DUP (0) ; es equivalente a DD 0,0,0,0 TABLA **DD** 4, ?, 650 ; no se define el segundo valor
DQ	**Definir cuádruple palabra** *Reserva memoria para una variable tipo doble palabra (64 bits)*	[variable] **DQ** expresión [,...]	VALORES **DQ** 300,-150,2000 MAXSS **DQ** 18446744073709551615 ; número máximo sin signo **DQ** 120 * 3 ; expresión de constantes **DQ** 4 DUP (0) ; es equivalente a DQ 0,0,0,0 TABLA **DQ** 4,?,650 ; no se define el segundo valor
DT	**Definir diez bytes** *Reserva diez bytes de memoria para almacenar dígitos decimales empaquetados (dos dígitos decimales por byte)*	[variable] **DT** expresión [,...]	NMAS **DT** 0123456789 ; reserva 10 bytes ; equivale a NMAS **DB** 00h,4 DUP(?),01h,23h,45h,67h,89h

- ***Control del Ensamblador***

DIRECTIVA	DESCRIPCIÓN	FORMATO	EJEMPLO
END	**Fin del módulo fuente** *El operando "expresión" indica la dirección de comienzo del programa fuente. Normalmente, se especifica una etiqueta.*	**END** [expresión]	**END** ; fin del modulo fuente **END** EMPEZAR ; fin del módulo fuente principal

- ***Definición de segmentos y procedimientos***

DIRECTIVA	DESCRIPCIÓN	FORMATO	EJEMPLO
SEGMENT	**Comienzo de segmento**	nombre **SEGMENT**	PILA **SEGMENT** STACK ; comienzo segmento PILA ... ... PILA **ENDS** ; fin segmento PILA
ENDS	**Fin de segmento o fin de estructura**	nombre **ENDS**	
ASSUME	**Suponer registro de segmento**	**ASSUME** reg_seg:nom_seg[,...] o **ASSUME NOTHING**	DATOS SEGMENT ; comienzo segmento DATOS ... ... DATOS ENDS ; fin segmento DATOS CODIGO SEGMENT ; comienzo segmento CODIGO **ASSUME** CS:CODIGO,DS:DATOS,ES:NOTHING ... ... CODIGO ENDS ; fin segmento CODIGO
PROC	**Comienzo de procedimiento**	nom_proc **PROC** [atributo]	PROC1 **PROC** FAR ; comienzo procedimiento ... ... RET PROC1 **ENDP** ; fin procedimiento
ENDP	**Fin de procedimiento**	nom_proc **ENDP**	

2- *DIRECTIVAS CONDICIONALES:*

DIRECTIVA	DESCRIPCIÓN	FORMATO	EJEMPLO
If **ELSE** **ENDIF**	Sirven para que el ensamblador incluya o ignore ciertas porciones del programa fuente, según que una cierta condición sea cierta o falsa en tiempo de ensamblaje.	**If** [condición] .... **[ELSE** ....] **ENDIF**	**IF** condicion1 .... **IF** condicion2 .... **ENDIF** ... ... **ENDIF**

3- *DIRECTIVAS DE LISTADO*: indican al ensamblador la información a obtener en el listado de salida y el formato de esa información.

- ***Formato de Listado***

DIRECTIVA	DESCRIPCIÓN	FORMATO	EJEMPLO
PAGE	**Formato de la página del listado**	**PAGE** [op1][,op2]	**PAGE** ; 66 líneas de 80 caracteres **PAGE** 88, 132 ; 88 líneas de 132 caracteres **PAGE** 88 ; 88 líneas de 80 caracteres **PAGE** ,132 ; 66 líneas de 132 caracteres **PAGE** + ; nuevo capítulo
TITLE	**Título del listado** *Especifica un título que aparecerá en el listado como primera línea en cada página*	**TITLE** texto	**TITLE** ESTRUCTURAS DE HORMIGON *El nombre del módulo objeto generado será* ESTRUC.OBJ
SUBTTL	**Subtítulo del listado** *Especifica un subtítulo que aparecerá en el listado como segunda línea en cada página, detrás del título.*	**SUBTTL** texto	**SUBTTL** CALCULO DE ESFUERZOS

- ***Comentarios***

DIRECTIVA	DESCRIPCIÓN	FORMATO	EJEMPLO
COMMENT	**Comentario** *Permite insertar comentarios en el programa sin tener que especificar el carácter ";" en cada línea*	**COMMENT** delimitador texto delimitador	**COMMENT** *Esto es un comentario que puede ocupar muchas líneas de texto. .... hasta que vuelva a aparecer el carácter inicial, en este caso, *