Ensamblador para el proyecto

<u>Características generales de la arquitectura</u>	
Propiedad	Valor
Unidad mínima direccionable (byte)	8 bits
Tamaño de la palabra	24 bits
Tamaño de una dirección	16 bits
Codificación de datos	Little endian (byte menos significativo primero)
Codificación de enteros	Complemento a dos

<u>Registros disponibles</u>	
Nombre del registro	Clave correspondiente en código máquina
R0	000
R1	001
R2	010
R3	011
R4	100
R5	101
RS (registro de estado o condición)	110
RP (registro de contador de programa)	111

<u>Sentencias del ensamblador</u>	
Sentencia	Descripción
OFFSET <i>entero</i>	Posicionamiento en la dirección de memoria indicada por <i>entero</i>
BYTE <i>nombre</i>	Apartar un byte bajo la etiqueta <i>nombre</i>
BYTE <i>nombre valor</i>	Apartar un byte bajo la etiqueta <i>nombre</i> e inicializarlo a <i>valor</i>
BYTE ARRAY <i>nombre tamaño</i>	Apartar un arreglo de <i>tamaño</i> bytes bajo la etiqueta <i>nombre</i>

WORD <i>nombre</i>	Apartar una palabra bajo la etiqueta <i>nombre</i>
WORD <i>nombre valor</i>	Apartar de una palabra bajo la etiqueta <i>nombre</i> e inicializarla a <i>valor</i>
WORD ARRAY <i>nombre tamaño</i>	Apartar un arreglo de <i>tamaño</i> palabras bajo la etiqueta <i>nombre</i>

Conjunto de instrucciones		
Instrucción	Código máquina	Descripción
NOP	00000000	Operación vacía
JMP <i>entero</i>	00000001 <i>eeeeeeee</i>	Salto incondicional (direccionamiento relativo a RP, $-128 \leq \text{entero} \leq 127$)
JIF <i>entero</i>	00000010 <i>eeeeeeee</i>	Salto condicional (direccionamiento relativo a RP, $-128 \leq \text{entero} \leq 127$)
JMP <i>entero</i>	00000011 <i>eeeeeeeeeeeeeeee</i>	Salto incondicional (direccionamiento absoluto)
JIF <i>entero</i>	00000100 <i>eeeeeeeeeeeeeeee</i>	Salto condicional (direccionamiento absoluto)
JMP <i>ra</i>	10000aaa	Salto incondicional (direccionamiento mediante registro)
JIF <i>ra</i>	10001aaa	Salto condicional (direccionamiento mediante registro)
MOV <i>ra, entero</i>	10010aaaa <i>eeeeeeee</i>	Asignación a registro (direccionamiento inmediato, $-128 \leq \text{entero} \leq 127$)
MOV <i>ra, entero</i>	10011aaaa <i>eeeeeeeeeeeeeeeeeeeeeeee</i>	Asignación a registro (direccionamiento inmediato)
MOV <i>ra, [entero]</i>	10100aaaa <i>eeeeeeeeeeeeeeee</i>	Lectura de una palabra en memoria (direccionamiento directo)
MOV <i>ra, BYTE [entero]</i>	10101aaaa <i>eeeeeeeeeeeeeeee</i>	Lectura de un byte en memoria (direccionamiento directo)
MOV [<i>entero</i>], <i>ra</i>	10110aaaa <i>eeeeeeeeeeeeeeee</i>	Escritura de una palabra en memoria (direccionamiento directo)
MOV BYTE [<i>entero</i>], <i>ra</i>	10111aaaa <i>eeeeeeeeeeeeeeee</i>	Escritura de un byte en memoria (direccionamiento directo)
MOV <i>ra, rb</i>	0100000000aaabbb	Transferencia entre registros
NEG <i>ra, rb</i>	0100000001aaabbb	Negación aritmética
COM <i>ra, rb</i>	0100000010aaabbb	Complemento a nivel bit
LST <i>ra, rb</i>	0100000011aaabbb	Menor que
LSE <i>ra, rb</i>	0100000100aaabbb	Menor o igual que
GRT <i>ra, rb</i>	0100000101aaabbb	Mayor que
GRE <i>ra, rb</i>	0100000110aaabbb	Mayor o igual que
EQU <i>ra, rb</i>	0100000111aaabbb	Igual que
NEQ <i>ra, rb</i>	0100001000aaabbb	Diferente que
MOV <i>ra, [rb]</i>	0100001001aaabbb	Lectura de una palabra en memoria (direccionamiento indirecto)

MOV <i>ra</i> , BYTE [<i>rb</i>]	0100001010aaabbbb	Lectura de un byte en memoria (direccionamiento indirecto)
MOV [<i>ra</i>], <i>rb</i>	0100001011aaabbbb	Escritura de una palabra en memoria (direccionamiento indirecto)
MOV BYTE [<i>ra</i>], <i>rb</i>	0100001100aaabbbb	Escritura de un byte en memoria (direccionamiento indirecto)
MOV <i>ra</i> , [<i>rb</i> + entero]	0100001101aaabbbbeeeeeeeeeeeeeee	Lectura de una palabra en memoria (base más desplazamiento)
MOV <i>ra</i> , BYTE [<i>rb</i> + entero]	0100001110aaabbbbeeeeeeeeeeeeeee	Lectura de un byte en memoria (base más desplazamiento)
MOV [<i>ra</i> + entero], <i>rb</i>	0100001111aaabbbbeeeeeeeeeeeeeee	Escritura de una palabra en memoria (base más desplazamiento)
MOV BYTE [<i>ra</i> + entero], <i>rb</i>	0100010000aaabbbbeeeeeeeeeeeeeee	Escritura de un byte en memoria (base más desplazamiento)
ADD <i>ra</i> , <i>rb</i> , <i>rc</i>	0010000aaabbbccc	Suma entre registros
SUB <i>ra</i> , <i>rb</i> , <i>rc</i>	0010001aaabbbccc	Resta entre registros
MUL <i>ra</i> , <i>rb</i> , <i>rc</i>	0010010aaabbbccc	Multiplicación entre registros
DIV <i>ra</i> , <i>rb</i> , <i>rc</i>	0010011aaabbbccc	División entre registros
MOD <i>ra</i> , <i>rb</i> , <i>rc</i>	0010100aaabbbccc	Módulo entre registros
AND <i>ra</i> , <i>rb</i> , <i>rc</i>	0010101aaabbbccc	AND a nivel bit entre registros
IOR <i>ra</i> , <i>rb</i> , <i>rc</i>	0010110aaabbbccc	OR inclusivo a nivel bit entre registros
XOR <i>ra</i> , <i>rb</i> , <i>rc</i>	0010111aaabbbccc	OR exclusivo a nivel bit entre registros
SHL <i>ra</i> , <i>rb</i> , <i>rc</i>	0011000aaabbbccc	Desplazamiento a la izquierda entre registros
SHR <i>ra</i> , <i>rb</i> , <i>rc</i>	0011001aaabbbccc	Desplazamiento a la derecha entre registros

Ejemplo de sintaxis y uso de etiquetas de ensamblador	
OFFSET 0	
WORD palabra 5	
OFFSET 10	
eti: MOV R0, [palabra]	# esto es un comentario, no es necesario implementarlos
JMP eti	