Ensambladores en la Actualidad

En la actualidad...

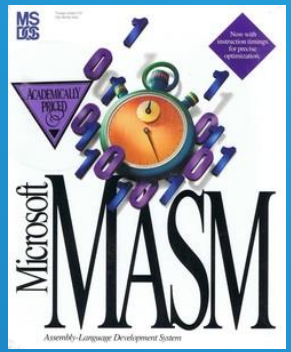
- El lenguaje ensamblador y los ensambladores han evolucionado conforme las necesidades y las implementaciones de hardware lo han hecho.

- Algunas iniciativas dieron como resultado:

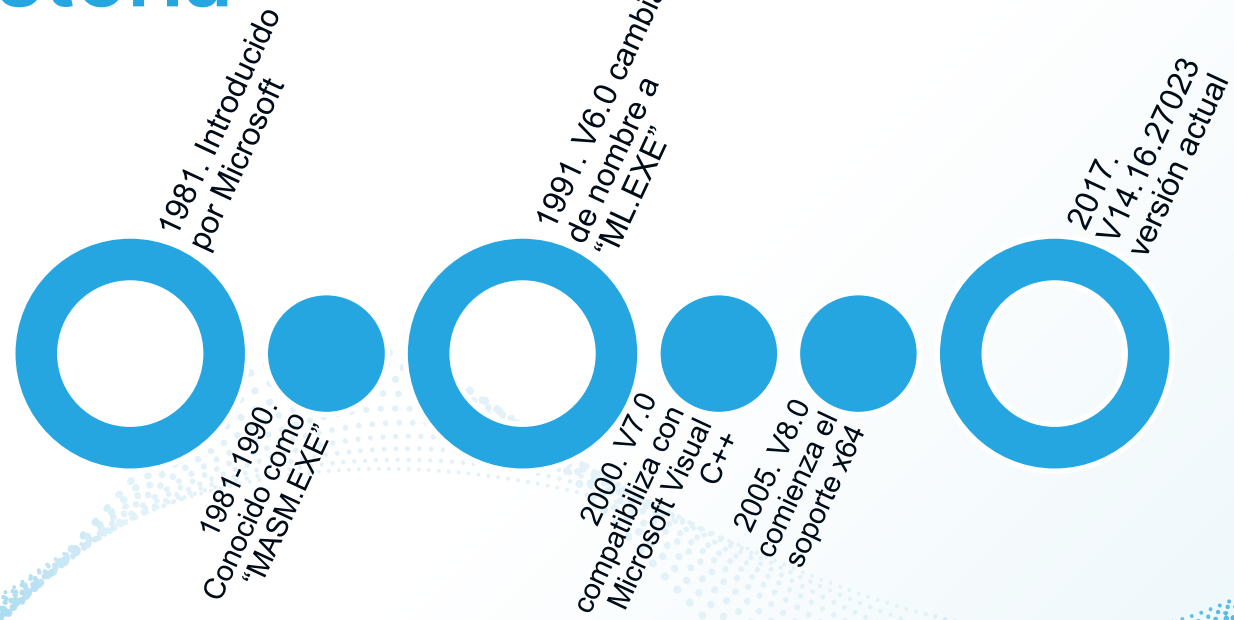
1. Macro Assembler (Microsoft)
2. RISC V (libre)

1.1 Macro Assembler

Historia



Historia

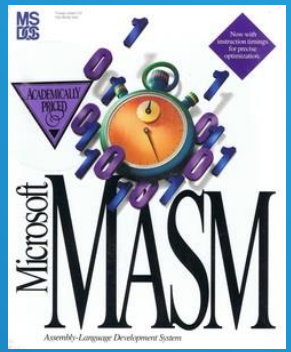

1981. Introducido por Microsoft
1981-1990. Conocido como "MASM.EXE"
1991. V6.0 cambia de nombre a "ML.EXE"
2000. V7.0 compatibiliza con Microsoft Visual C++
2005. V8.0 comienza el soporte x64
2017. V14.16.27023 versión actual

Historia

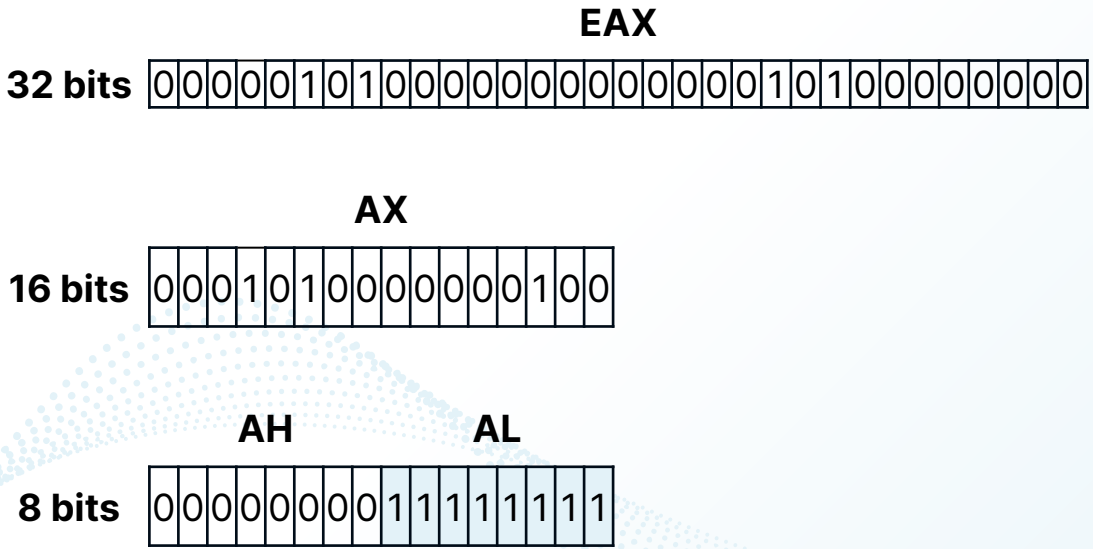
- Los procesadores Intel 8086 – 80286 permitían programas con datos 8 – 16 bits (Turbo Assembler, etc.)
- El procesador Intel 80386 fue el primer procesador en permitir programas con datos de 8, 16, 32 bits, con set de instrucciones x86 (Macro Assembler, etc.)
- A partir de Windows NT las palabras convierten su tamaño oficial a 32 bits, comienza el set de instrucciones x64.

1.2 Macro Assembler

Registros



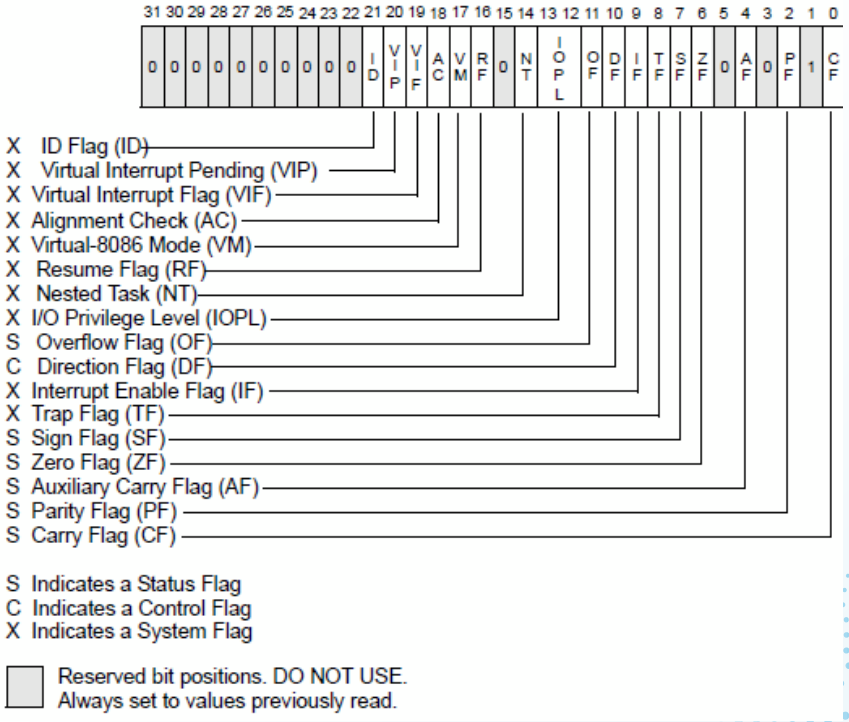
Registros


EAX
32 bits
0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0
AX
16 bits
0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0
AH
AL
8 bits
0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1

Registros

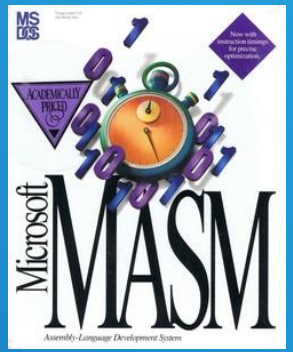
EAX	• Acumulador para operandos y resultados.
EBX	• Apuntador base para datos del segmento de datos.
ECX	• Contador para los ciclos.
EDX	• Apuntador para Entradas/Salidas
EBP	• Frame Pointer, utilizado para operaciones con la pila.
ESP	• Stack Pointer, utilizado en operaciones de PUSH y POP.
ESI	• Source Index, requerido para algunas operaciones con arreglos.
EDI	• Data Index, requerido para algunas operaciones con arreglos.
EIP	• Instruction Pointer, guarda la dirección de la siguiente instrucción a ejecutar.
EFLAGS	• Resultados de banderas.

EFLAGS


31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
0 0 0 0 0 0 0 0 0 0 ID VIP VIF AC VM RF 0 NT IOPL OF DF IF TF SF ZF 0 AF 0 PF 1 CF
X ID Flag (ID)
X Virtual Interrupt Pending (VIP)
X Virtual Interrupt Flag (VIF)
X Alignment Check (AC)
X Virtual-8086 Mode (VM)
X Resume Flag (RF)
X Nested Task (NT)
X I/O Privilege Level (IOPL)
S Overflow Flag (OF)
C Direction Flag (DF)
X Interrupt Enable Flag (IF)
X Trap Flag (TF)
S Sign Flag (SF)
S Zero Flag (ZF)
S Auxiliary Carry Flag (AF)
S Parity Flag (PF)
S Carry Flag (CF)
S Indicates a Status Flag
C Indicates a Control Flag
X Indicates a System Flag
Reserved bit positions. DO NOT USE.
Always set to values previously read.

1.3 Macro Assembler

Modelos



Modelos

Tiny	Datos y código en el mismo segmento. **64K bytes en total**
Small	Datos y código en segmentos independientes. **64K bytes c/u**
Compact	Datos crean nuevos segmentos al llenar, código en segmento único. **64K bytes c/u**
Medium	Datos en segmento único, código crea nuevos segmentos al llenar. **64K bytes c/u**
Large	Datos y código crean nuevos segmentos al llenar. **64K bytes c/u**
Huge	Esencialmente igual a Large, pero permite manejo de datos mayores a un segmento. Lo define el programador. **64K bytes c/u**
Flat	Datos y código en el mismo segmento. **Segmento 32-bits**

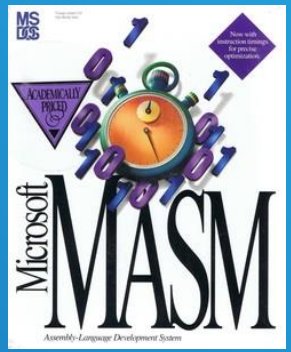
Modelo

.MODEL ***memory-model*** ⟦**,** ***language-type***⟧ ⟦**,** ***stack-option***⟧

Parámetro	32-bit	16-bit
memory-model	FLAT	TINY, SMALL, COMPACT, MEDIUM, LARGE, HUGE, FLAT
language-type	C, STDCALL	C, BASIC, FORTRAN, PASCAL, SYSCALL, STDCALL
stack-option	Not used	NEARSTACK, FARSTACK

1.4 Macro Assembler

Consideraciones



Consideraciones

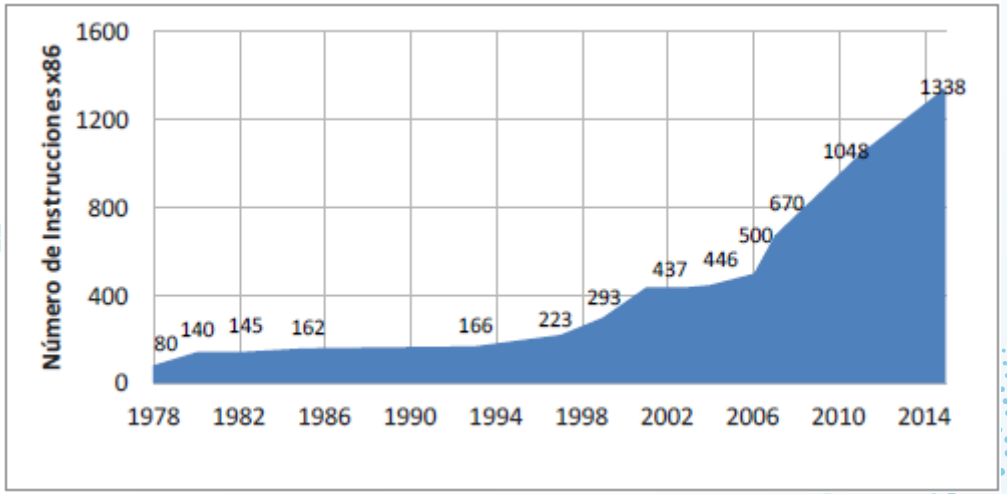
- Debe definirse el procesador con el que se trabajará (.386)
- Por defecto el sistema numérico es hexadecimal, pero se puede modificar.
- La pila (.STACK) por defecto el tamaño es de 1024 bytes, pero se puede modificar.
- Segmento de datos (.DATA) y datos no inicializados (.DATA?)

2.1 RISC V

Historia

Historia

- Las ISA existentes eran complejas y con derechos de propiedad intelectual.
- Partiendo del Intel 8086, el set x86 parte con 80 instrucciones en 1978; para el 2014 alcanzó las 1338 en su versión x86 32 bits.



Historia

- RISC (Reduced Instruction Set Computer) inició como proyecto temporal en UC Berkeley.
- Surge de la necesidad de una ISA libre y cambiando la tendencia incremental de las ISAs existentes a una modular, con un núcleo fundamental y adaptándolo a las necesidades específicas.
- El núcleo fundamental del ISA de RISC-V es llamado RV32I.
- Las extensiones se indican mediante banderas representadas mediante concatenación al núcleo fundamental (ejemplo RV32IMF, agrega multiplicación RV32M y punto flotante precisión simple RV32F, a las instrucciones base obligatorias RV32I).

¿Dudas?