

# Einführung in Reverse Engineering

---

Martin Morgenstern

14.04.2023

# Kurzwiederholung x86-Assembler

- Assembler allgemein = Programmiersprache zur Steuerung einer Rechners → Abhängig vom Prozessor-Typ
- ABER: Assembler wird auch das Programm zur Übersetzung genannt
- X86 bzw. x86\_64 ist die am meisten verbreitet Rechnerarchitektur bei Laptops, Desktops-PCs und Servern
- Grundverständnis von Assembler ist notwendig für Reverse-Engineering; aber Sie müssen nicht alle Befehle können

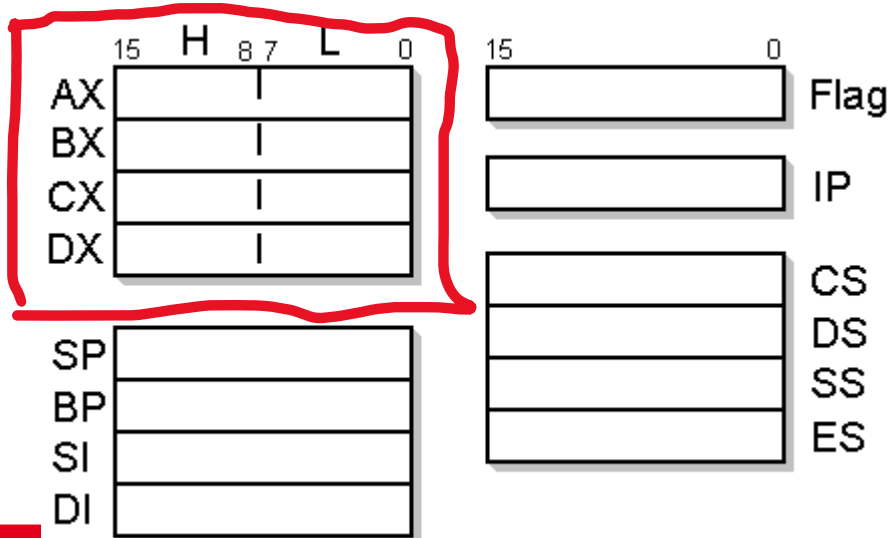
# Register 8086 und Nachfolger

- 8086 hatte nur 16Bit, die Architektur und Funktionsweise, sowie die Namensgebung der Register ist jedoch bis heute identisch  
→ auch auf modernen x86-CPU's können alte Betriebssysteme, z. B. MS DOS, ausgeführt werden
- Allgemeine Einteilung: Datenregister, Zeiger- und Indexregister
- Weitere Details siehe <https://www.i8086.de/register/register.html>

# Datenregister des 8086

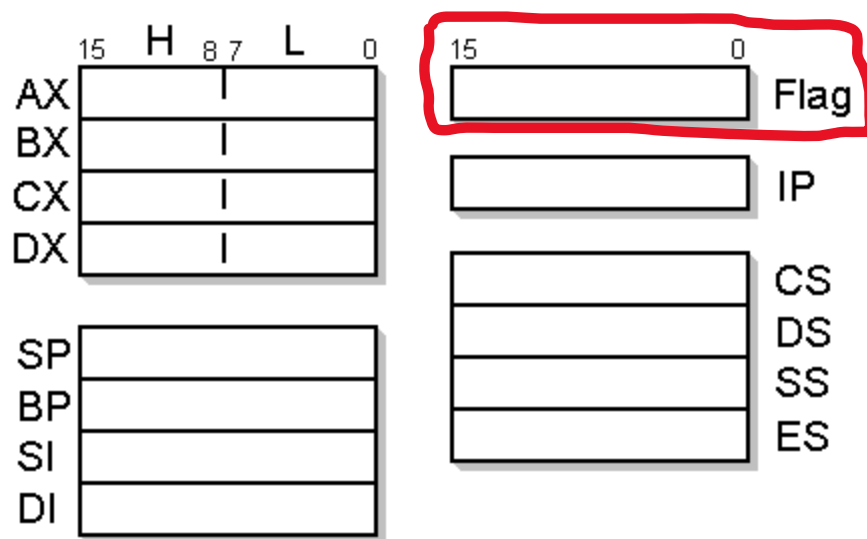
- AX (Akkumulator), BX (Basis), CX (Zähler), DX (Daten)
- → Aufteilung der Register jeweils in High (H) und Low (L) siehe Grafik → Es ist möglich Teilregister direkt anzusprechen
- Relativ freie Verwendung der Register möglich (CX für Loops erwartet)

• Grafik von [https://de.m.wikibooks.org/wiki/Datei:Register\\_8086.PNG](https://de.m.wikibooks.org/wiki/Datei:Register_8086.PNG) von User „Daniel B“



# Flagregister des 8086

- Enthält die Flags, also Zustandswerte (siehe Folie Flags)
- Grafik von [https://de.m.wikibooks.org/wiki/Datei:Register\\_8086.PNG](https://de.m.wikibooks.org/wiki/Datei:Register_8086.PNG) von User „Daniel B“



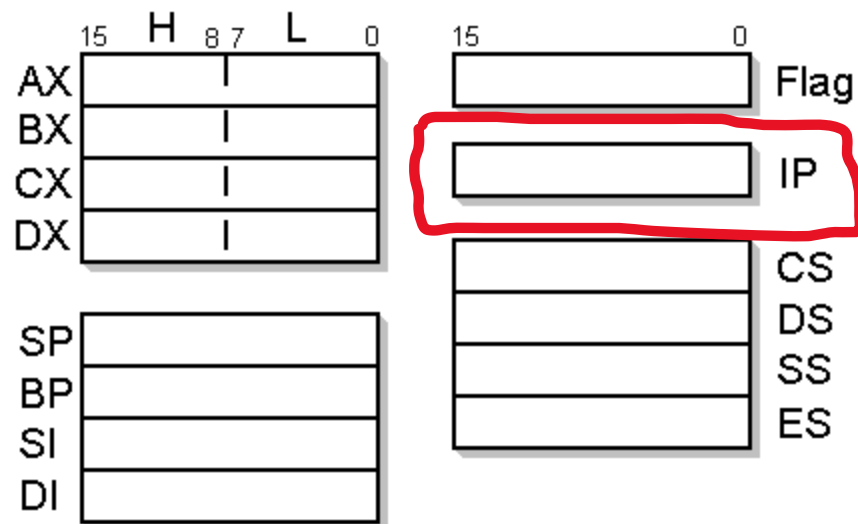
# Flag-Register

Intel x86 FLAGS register <sup>[1]</sup>			
Bit #	Abbreviation	Description	Category
FLAGS			
0	CF	Carry flag	Status
1		Reserved, always 1 in EFLAGS <sup>[2]</sup>	
2	PF	Parity flag	Status
3		Reserved	
4	AF	Adjust flag	Status
5		Reserved	
6	ZF	Zero flag	Status
7	SF	Sign flag	Status
8	TF	Trap flag (single step)	Control
9	IF	Interrupt enable flag	Control
10	DF	Direction flag	Control
11	OF	Overflow flag	Status
12-13	IOPL	I/O privilege level (286+ only), always 1 on 8086 and 186	System
14	NT	Nested task flag (286+ only), always 1 on 8086 and 186	System
15		Reserved, always 1 on 8086 and 186, always 0 on later models	

# Instruction Pointer (IP) des 8086

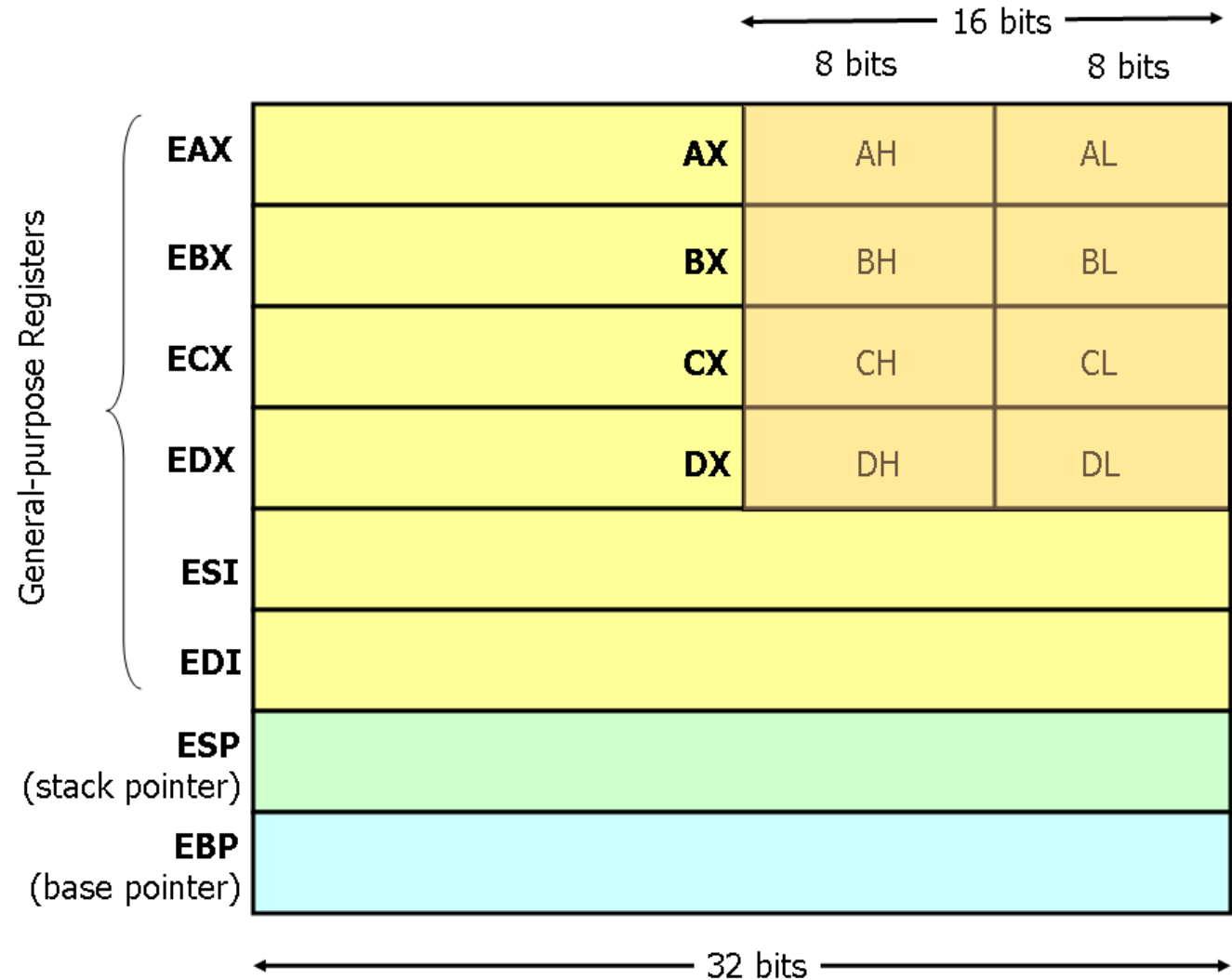
- Enthält die Adresse des nächsten auszuführenden Befehls

- Grafik von [https://de.m.wikibooks.org/wiki/Datei:Register\\_8086.PNG](https://de.m.wikibooks.org/wiki/Datei:Register_8086.PNG) von User „Daniel B“



# 32-Bit Erweiterung

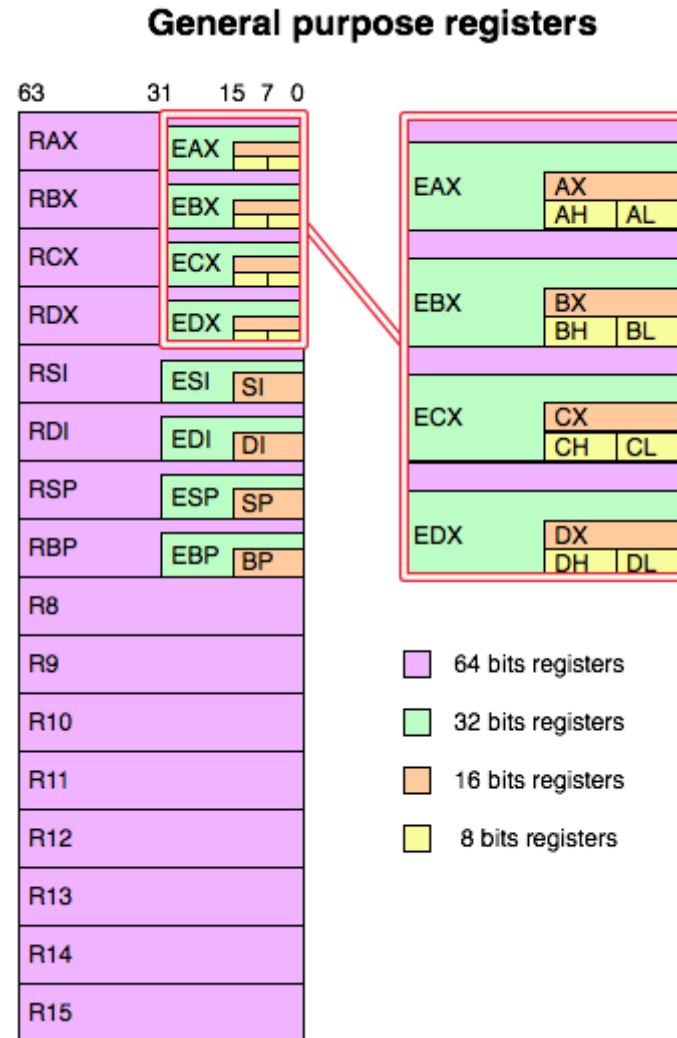
- Register enthalten alte Namen, werden jedoch mit einem E (für Extended)
- Grafik von Ipay Yildirim  
<https://github.com/AYIDouble/x86-Assembly-Reverse-Engineering/blob/master/Images/x86-registers.png>





# 64-Bit Erweiterung

- Register enthalten alte Namen, werden jedoch mit einem E (für Extended)
- Zusätzliche Register R8 bis 15
- Grafik von <https://clementbera.wordpress.com/2014/03/11/intel-registers/>



# Assembler-Varianten

- Assembler != Assembler; Grundfunktionen sind gleich, jedoch unterscheidet sich die Syntax oft
- Beispiel mov-Befehl bei MASM und FASM
  - MASM "*mov destination, source*"-Format
  - FASM "*mov source, destination*"-Format

# Assembler Syntaxe

- Verschiedene Syntaxe sind möglich; gebräuchlich sind die Intel- und die AT&T-Syntax
- Intel-Syntax: besser lesbar
- Oft kann die Syntax umgestellt werden

# Assembler-Befehle (x86)

- Überschaubarer Befehlssatz im Vergleich zu höheren Programmiersprachen
- Auflistung unter [https://de.wikibooks.org/wiki/Assembler-Programmierung\\_f%C3%BCr\\_x86-Prozessoren/Befehlsliste](https://de.wikibooks.org/wiki/Assembler-Programmierung_f%C3%BCr_x86-Prozessoren/Befehlsliste)

# Befehlsarten in Assembler

- Datentransferbefehle (z. B. `mov eax, 1`)
- Arithmetische Befehle (z. B. `add`, `sub`, `and`)
- Sprungbefehle (diverse Jump-Befehle z.B. `jmp`, `je`, `jne` )
- Stapelbefehle (`push`, `pop`)
- Kontrollbefehle (z. B. `cmp` )
- Weitere Befehle (z. B. `nop` )

Im Internet finden Sie genaue Erklärungen und Beispiele

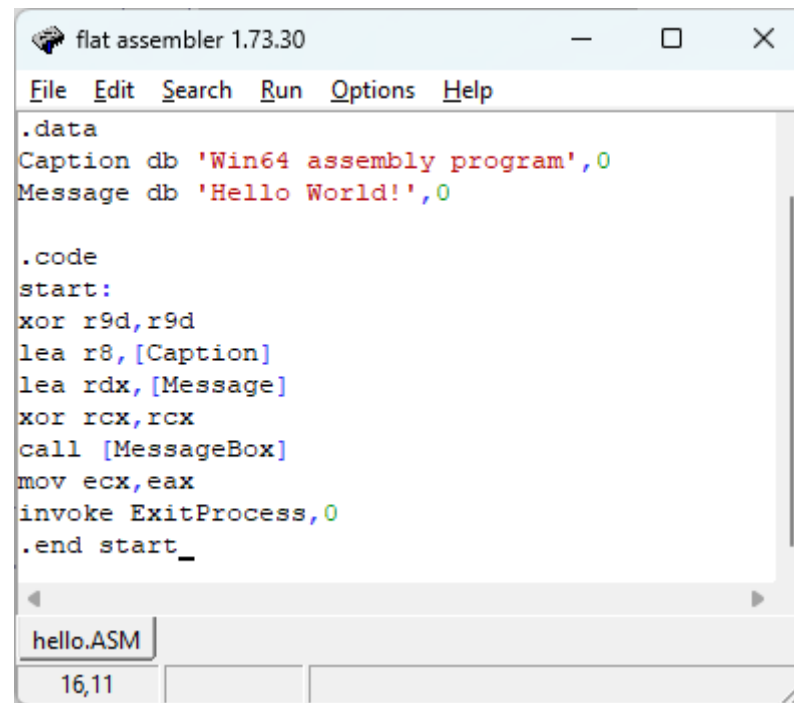
# Flags

- Flags sind Status-Informationen
- Bedeutung z. B. für Rechenoperationen und Ergebnisse von Vergleichen

# Flat Assembler

- klein und schlank
- betriebssystemübergreifend
- Open-Source
- <https://flatassembler.net/download.php>
- Beispiele für einfache Programme  
<https://flatassembler.net/examples.php>

# Einfaches Hello-World in Flat Assembler



```
flat assembler 1.73.30
File Edit Search Run Options Help

.data
Caption db 'Win64 assembly program',0
Message db 'Hello World!',0

.code
start:
xor r9d,r9d
lea r8,[Caption]
lea rdx,[Message]
xor rcx,rcx
call [MessageBox]
mov ecx,eax
invoke ExitProcess,0
.end start_

hello.ASM
16,11
```



# IDA Pro

- Kostenlos (mit eingeschränkten Funktionsumfang) für nicht-kommerzielle Zwecke
  - Quasi-Standard im RE unter Windows (Linux-Version verfügbar)
  - Für kommerzielle Zwecke kostenintensiv
- Funktionen werden in Live-Demo gezeigt

# Organisatorisches Prüfung

- Einige Prüfungsaufgaben sind bereits fertig, aber nicht alle
- Vorschlag: die Prüfungsaufgaben die bereits fertig sind werden zuerst ausgegeben → dann auch zuerst Vortrag
- Jeder erhält die gleiche Zeit zur Prüfungsbearbeitung

# Testversion von MSAB

Wir haben für ca. 3 Wochen eine Testversion von MSAB zur Auswertung und Analyse von Smartphones

Vorschlag: Die Gruppe mit Smartphone-Auswertung beschäftigt ist erhält diese, dafür soll es ein Teil der Prüfung das Tool zu testen und vorzustellen.

# Radare2

- OpenSource
- In vielen Linux-Distributionen bereits enthalten