Organisasi Sistem Komputer

Bab 2. Pengenalan ke Program Assembly (NASM)

2.1 Arsitektur x86

2.2 Assembler dan Linker

2.3 Menulis Hello World dalam Bahasa Assembly

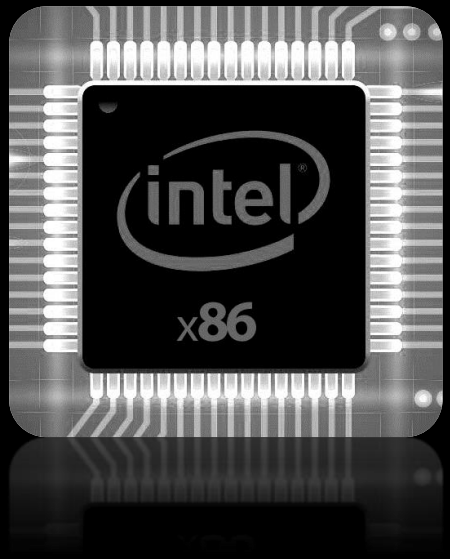
2.4 Struktur Program NASM

Pembahasan:

- Arsitektur x86
- General Purpose Register
- Register EIP dan EFLAGS
- ISA x86

Arsitektur x86

- Untuk mempelajari Bahasa assembly kita perlu memilih satu keluarga prosesor, karena instruksi-instruksi assembly berbeda untuk keluarga prosesor yang berbeda
- Kita menggunakan bahasa assembly untuk keluarga prosesor 32-bit Intel 80x86 (x86 singkatnya)
- Untuk menulis program assembly kita perlu memahami:
  - Register-register yang tersedia
  - Instruction Set Architecture (ISA)

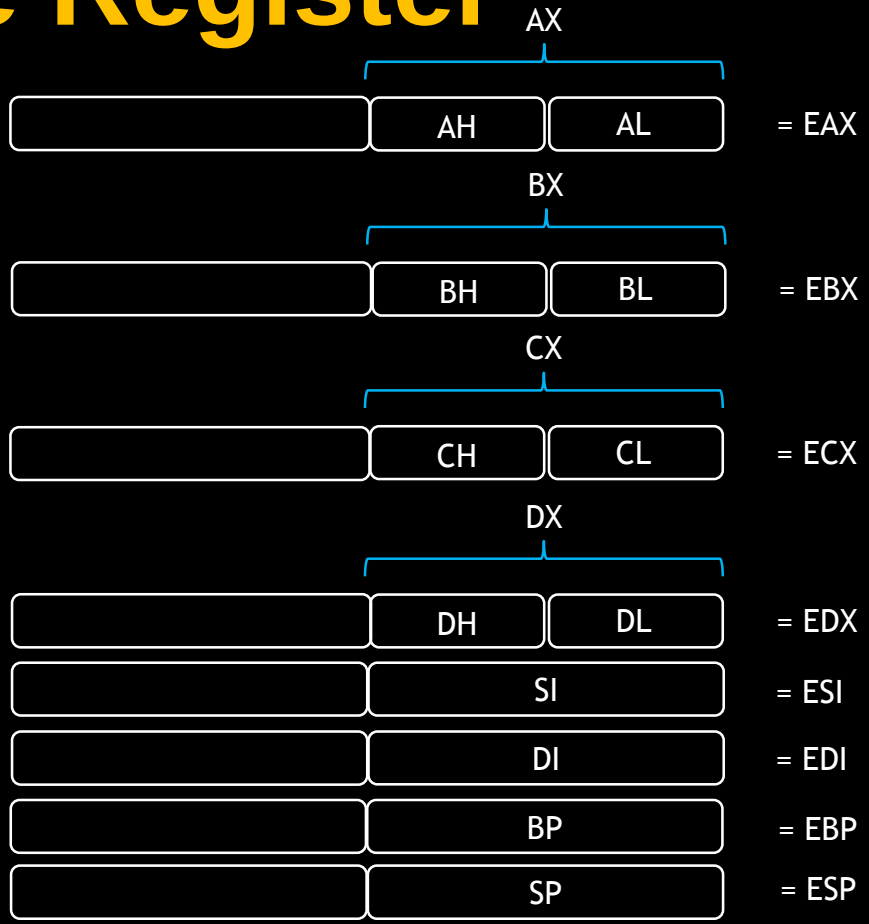


Register x86

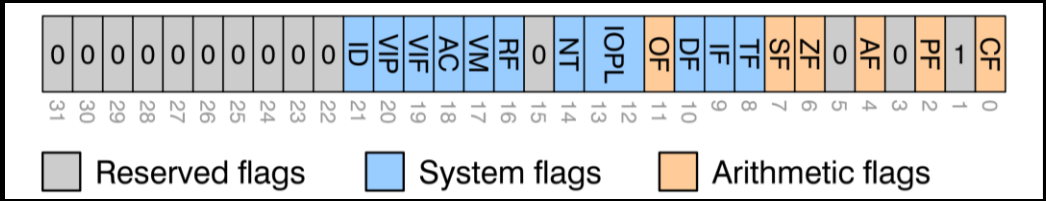
- Register adalah tempat penyimpanan sementara dalam prosesor
- Terdapat banyak register pada x86, namun untuk pemrograman dasar assembly, kita hanya memerlukan register dalam tiga kategori berikut:
  - General Purpose Register:
    - Terdiri dari 8 register 32-bit
    - Digunakan untuk tempat penyimpanan sementara sebelum operasi aritmatika, operasi bit, dsb.
  - Intruction Pointer:
    - Satu register 32-bit: EIP (Extended Instruction Pointer)
    - Menyimpan alamat memori dari instruksi berikutnya yang akan dieksekusi
    - Nilainya dapat kita manipulasi untuk mengubah alur program (percabangan)
  - Register EFLAGS:
    - Satu register 32-bit: EFLAGS (Extended Flags)
    - Setiap bit-nya mempunyai arti tertentu
    - Digunakan untuk melihat status hasil aritmatika dan perbandingan

General Purpose Register

- Terdapat 8 register 32-bit: EAX, EBX, ECX, EDX, ESI, EDI, EBP, dan ESP
- EAX, EBX, ECX, dan EDX dapat digunakan 16 bit terendahnya: AX, BX, CX, DX
- AX, BX, CX, dan DX dapat digunakan:
  - 8 bit terendahnya: AL, BL, CL, DL
  - 8 bit tertingginya: AH, BH, CH, DH
- ESI, EDI, EBP, dan ESP dapat digunakan 16 bit terendahnya: SI, DI, BP, SP
- ESP (Extended Stack Pointer) dan EBP (Extended Base Pointer) mempunyai kegunaan khusus saat menggunakan stack



Register EFLAGS



- **Carry Flag (CF)** – menandakan hasil operasi aritmatika bilangan tidak bertanda melebihi jangkauan ukuran bit tujuan (bernilai 1 jika ya)
- **Overflow Flag (OF)** – menandakan hasil operasi aritmatika bilangan bertanda melebihi jangkauan ukuran bit tujuan (bernilai 1 jika ya)
- **Sign Flag (SF)** – menandakan jika hasil operasi aritmatika menghasilkan nilai negatif (bernilai 1 jika ya)
- **Zero Flag (ZF)** – menandakan jika hasil operasi aritmatika menghasilkan nilai nol (bernilai 1 jika ya)
- **Auxiliary Carry Flag (AF)** – menandakan jika hasil operasi aritmatika menyebabkan sebuah bit carry dari bit 3 ke bit 4 dalam operand 8 bit (bernilai 1 jika ya)
- **Parity Flag (PF)** – bernilai 1 ketika jumlah bit-bit pada hasil operasi aritmatika adalah genap

ISA x86

- Spesifikasi ISA untuk Intel x86 dapat dilihat di http://ref.x86asm.net/

pf	0F	po	so	o	proc	st	m	rl	x	mnemonic	op1	op2	op3	op4	iext	tested f	modif f	def f	undef f	f values	description, notes
		00		r					L	ADD	**r/m8**	r8					o..szapc	o..szapc			Add
		01		r					L	ADD	**r/m16/32**	r16/32					o..szapc	o..szapc			Add
		02		r						ADD	**r8**	r/m8					o..szapc	o..szapc			Add
		03		r						ADD	**r16/32**	r/m16/32					o..szapc	o..szapc			Add
		04								ADD	**AL**	imm8					o..szapc	o..szapc			Add
		05								ADD	**eAX**	imm16/32					o..szapc	o..szapc			Add
		06								PUSH	ES										Push Word, Doubleword or Quadword Ont
		07								POP	**ES**										Pop a Value from the Stack
		08		r					L	OR	**r/m8**	r8					o..szapc	o..sz.pc	.....a..	o......c	Logical Inclusive OR
		09		r					L	OR	**r/m16/32**	r16/32					o..szapc	o..sz.pc	.....a..	o......c	Logical Inclusive OR
		0A		r						OR	**r8**	r/m8					o..szapc	o..sz.pc	.....a..	o......c	Logical Inclusive OR
		0B		r						OR	**r16/32**	r/m16/32					o..szapc	o..sz.pc	.....a..	o......c	Logical Inclusive OR
		0C								OR	**AL**	imm8					o..szapc	o..sz.pc	.....a..	o......c	Logical Inclusive OR
		0D								OR	**eAX**	imm16/32					o..szapc	o..sz.pc	.....a..	o......c	Logical Inclusive OR
		0E								PUSH	ES										Push Word, Doubleword or Quadword Ont
		0F			02+					*Two-byte*		*tructions*									

operands

opcode dalam Hex

mnemonic

Ringkasan

- Untuk memulai menulis kode assembly kita perlu menentukan arsitektur prosesor apa yang menjadi tujuan kode assembly kita
- Kita perlu mengetahui register-register yang tersedia terutama:
  - General Purpose Register
  - Register Instruction Pointer
  - Register Flag
- Kita juga perlu mengetahui instruksi-instruksi apa yang tersedia pada Instruction Set Architecture (ISA)