Arsitektur von Neumann dan Harvard, Alur data dan Kontrol

Q Fadlan, S.ST, M.Kom

23 Oktober 2024

Q Fadlan, S.ST, M.Kom

Formal Education :

- D4 Teknik Telekomunikasi - PENS ITS (2012)
- S2 Magister Teknologi Komputer - Swiss German University (2018)

Currently work in one of Financial Technology in Indonesia as Information Security Manager

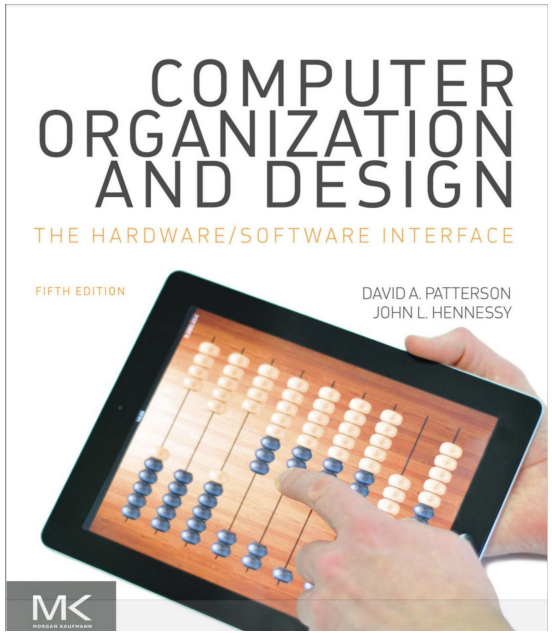
WA : 085294185488

Email : qfadlan@stmik.tazkia.ac.id

AGENDA

- Arsitektur von Neumann dan Arsitektur Harvard
- Perbedaan Arsitektur von Neumann dan Harvard
- Alur Data (Datapath)
- Control Unit
- Praktik Menggunakan *Little Man Computer* (LMC)

Referensi



Judul Buku : Computer Organization and Design: The Hardware/Software Interface
Penulis : David A. Patterson & John L. Hennessy

Arsitektur von Neumann

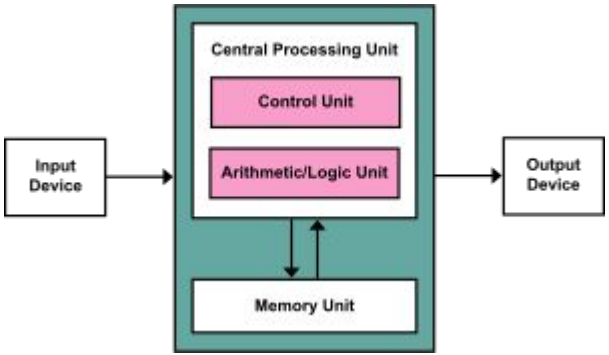
Arsitektur von Neumann adalah desain dasar untuk komputer elektronik yang dikembangkan oleh matematikawan John von Neumann dan rekan-rekannya pada tahun 1940-an. Arsitektur ini menjadi dasar bagi sebagian besar komputer modern.

Arsitektur von Neumann adalah sebuah arsitektur komputer dimana unit memori menyimpan instruksi program dan data. Komputer yang menggunakan arsitektur ini memiliki unit pemrosesan pusat (CPU) yang terdiri dari unit aritmatika dan logika (ALU), control unit (CU), dan register, bersama dengan perangkat input dan output (I/O).

Arsitektur von Neumann

Karakteristik utama arsitektur von Neumann meliputi:

1. Memori tunggal yang menyimpan program dan data
2. Unit pemrosesan pusat (CPU) untuk mengeksekusi instruksi
3. Unit aritmatika dan logika (ALU) untuk operasi matematika dan logika
4. Unit kontrol untuk mengelola urutan eksekusi instruksi
5. Perangkat input dan output untuk interaksi dengan dunia luar



Source image : https://en.wikipedia.org/wiki/Von_Neumann_architecture

Arsitektur Harvard

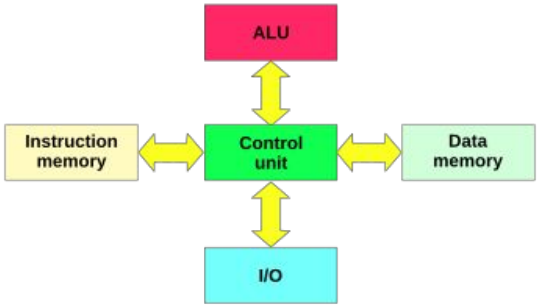
Arsitektur Harvard adalah desain arsitektur komputer yang dikembangkan sebagai alternatif dari arsitektur von Neumann. Nama "Harvard" berasal dari Harvard Mark I, komputer elektro-mekanik yang menggunakan desain ini.

Arsitektur Harvard adalah sebuah rancangan komputer di mana ada pemisahan fisik antara penyimpanan dan jalur sinyal untuk instruksi dan data. Ini berarti komputer memiliki memori terpisah dan bus terpisah untuk instruksi program dan data.

Arsitektur Harvard

Karakteristik utama arsitektur Harvard meliputi:

1. Memori terpisah untuk program dan data
2. Bus terpisah untuk mengakses instruksi dan data
3. Kemampuan untuk mengakses instruksi dan data secara bersamaan
4. Potensi untuk optimasi yang lebih baik dalam hal kecepatan dan keamanan



Source Image : https://en.wikipedia.org/wiki/Harvard_architecture

Perbedaan Arsitektur von Neumann dan Harvard

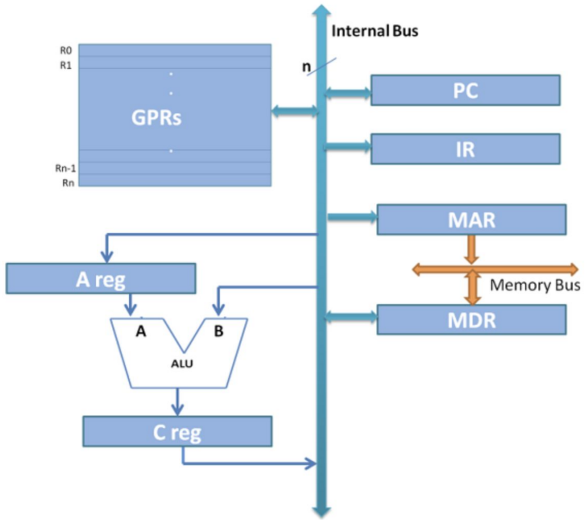
	von Neumann	Harvard
Struktur Memori	Menggunakan satu memori tunggal untuk menyimpan baik instruksi program maupun data	Memiliki memori terpisah untuk instruksi program dan data
Bus Sistem	Menggunakan satu set bus untuk mentransfer baik instruksi maupun data antara CPU dan memori	Memiliki bus terpisah untuk instruksi dan data, memungkinkan transfer simultan
Akses Simultan	Tidak dapat mengakses instruksi dan data secara bersamaan karena menggunakan bus tunggal	Dapat mengakses instruksi dan data secara bersamaan karena memiliki bus terpisah.
Kompleksitas Hardware	Umumnya lebih sederhana dalam desain hardware	Lebih kompleks karena memerlukan dua set memori dan bus terpisah

Alur Data (Datapath)

Alur data (data path) adalah kumpulan unit-unit fungsional seperti Arithmetic Logic Units (ALUs), register, dan bus yang melakukan operasi pemrosesan data dalam komputer.

Kenapa harus mengerti alur data?

- Alur data adalah inti dari cara CPU bekerja. Memahaminya berarti memahami bagaimana komputer sebenarnya memproses instruksi dan data.
- Memberikan wawasan tentang bagaimana setiap komponen dalam CPU berinteraksi untuk menyelesaikan tugas komputasi.



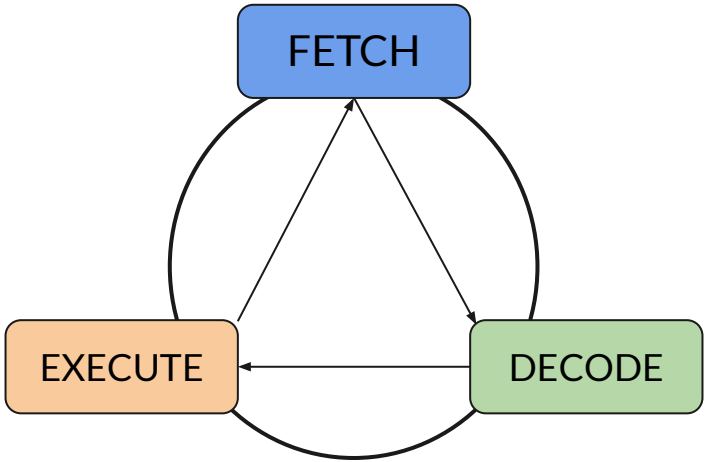
Source image :
https://witscad.com/course/computer-architecture/chapter/cpu-data-path

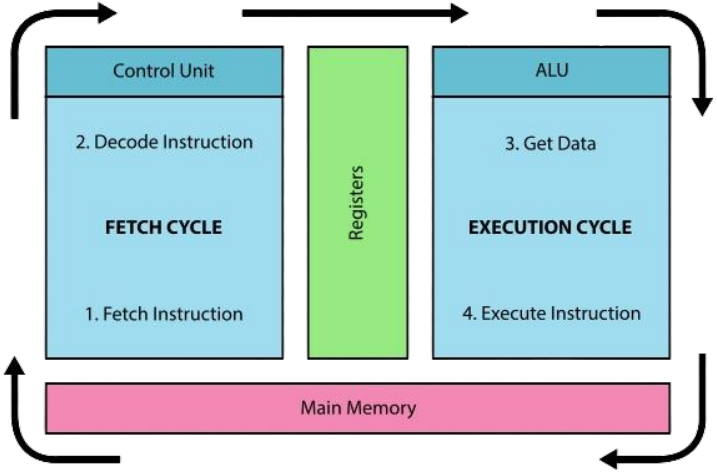
Alur Data (Datapath)

Komponen Utama Datapath:

- **Register**: Unit penyimpanan data sementara yang sangat cepat. Register digunakan untuk menyimpan data yang sedang diproses, seperti hasil perhitungan atau alamat memori.
- **Arithmetic Logic Unit (ALU)**: Unit yang melakukan operasi aritmatika (penjumlahan, pengurangan, perkalian, pembagian) dan logika (AND, OR, NOT).
- **Interkoneksi**: Kumpulan bus atau jalur yang menghubungkan komponen-komponen datapath, memungkinkan data mengalir antar komponen.

Alur Data (Datapath)



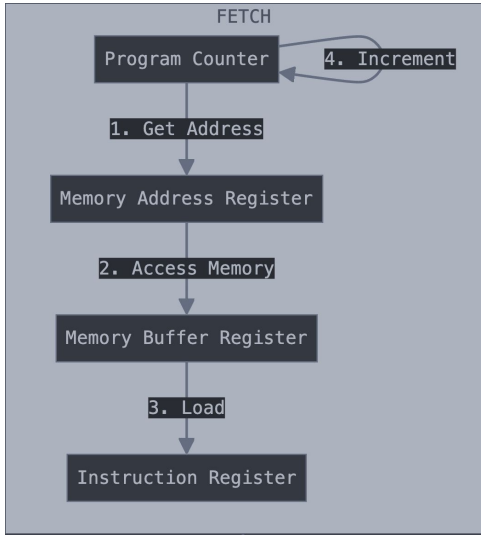


Fetch-Decode-Execute cycle

Alur Data (Datapath)

FETCH (Pengambilan) Definisi: Tahap dimana CPU mengambil instruksi dari memori utama. Komponen Utama:

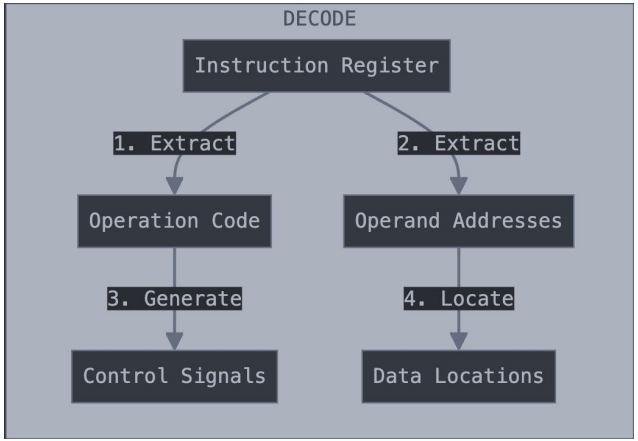
- Program Counter (PC): Register yang menyimpan alamat instruksi berikutnya
- Memory Address Register (MAR): Menyimpan alamat memori yang akan diakses
- Memory Buffer Register (MBR): Menyimpan data yang dibaca dari memori
- Instruction Register (IR): Menyimpan instruksi yang sedang diproses



Alur Data (Datapath)

DECODE (Penguraian) Definisi: Tahap dimana instruksi diinterpretasikan dan diurai menjadi bagian-bagian yang dapat dimengerti CPU. Komponen Utama:

- Instruction Decoder: Mengurai instruksi menjadi Operation Code (opcode) dan operand
- Control Unit: Menghasilkan sinyal kontrol berdasarkan opcode
- Register Decoder: Mengidentifikasi register yang terlibat



Alur Data (Datapath)

Opcode

- **Definisi:** Opcode adalah singkatan dari operation code. Ini adalah bagian dari instruksi yang menentukan **operasi** apa yang akan dilakukan oleh CPU.
- **Fungsi:** Opcode memberikan "perintah" kepada CPU tentang tindakan yang harus dilakukan, seperti penjumlahan, pengurangan, perbandingan, pemindahan data, dan sebagainya
- **Contoh:** Dalam instruksi penjumlahan, opcode akan menunjukkan bahwa operasi yang akan dilakukan adalah penjumlahan.

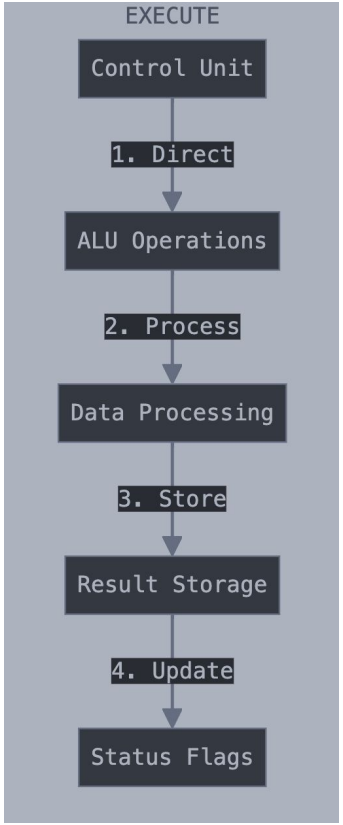
Operand

- **Definisi:** Operand adalah data yang akan dioperasikan oleh opcode.
- **Fungsi:** Operand memberikan input untuk operasi yang ditentukan oleh opcode
- **Contoh:** Dalam instruksi penjumlahan, operand adalah dua bilangan yang akan dijumlahkan.

Alur Data (Datapath)

EXECUTE (Eksekusi) Definisi: Tahap dimana instruksi yang telah didecode dijalankan oleh CPU. Komponen Utama:

- Arithmetic Logic Unit (ALU): Melakukan operasi aritmatika dan logika
- Registers: Menyimpan data temporari dan hasil
- Control Unit: Mengontrol aliran data dan timing



Control Unit

Control Unit adalah unit yang bertanggung jawab untuk mengatur, mengoordinasikan, dan mengontrol semua operasi yang terjadi di dalam CPU melalui serangkaian sinyal kontrol yang dihasilkan berdasarkan instruksi yang sedang dieksekusi.

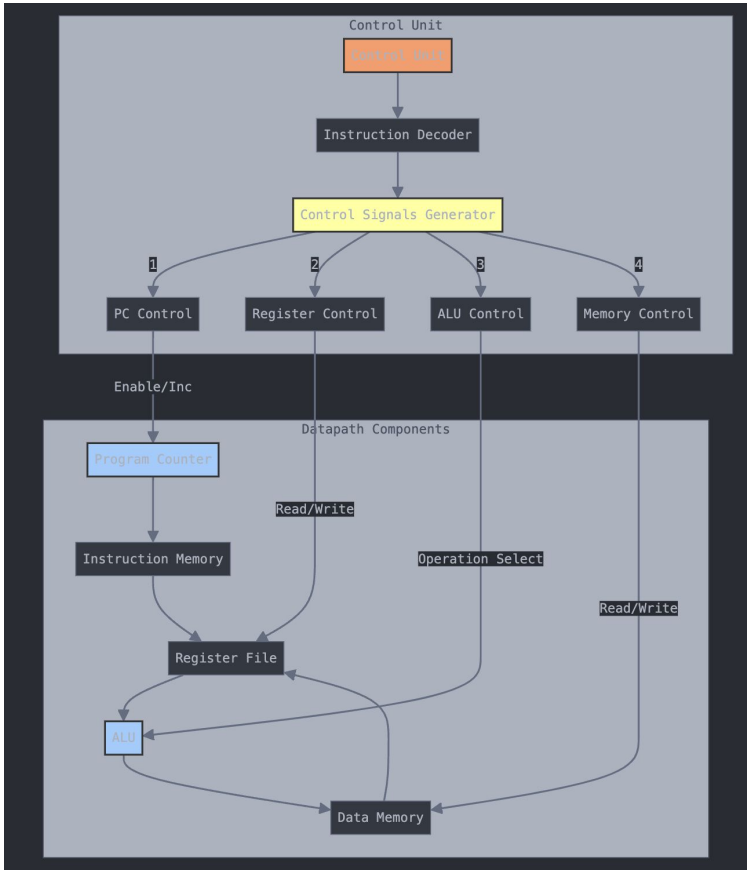
Control Unit

- **Mendekode Instruksi:** Ketika sebuah instruksi diambil dari memori, control unit akan menganalisis instruksi tersebut untuk menentukan operasi apa yang akan dilakukan (misalnya, penjumlahan, perbandingan, pemindahan data) dan pada data apa operasi tersebut akan dilakukan.
- **Mengirimkan Sinyal Kontrol:** Berdasarkan hasil dari proses decode, control unit akan mengirimkan sinyal-sinyal kontrol ke berbagai komponen datapath. Sinyal-sinyal ini akan mengaktifkan atau menonaktifkan komponen-komponen tertentu dan menentukan bagaimana data akan mengalir di dalam datapath.
- **Mengatur Urutan Operasi:** Control unit menentukan urutan di mana operasi-operasi harus dilakukan. Misalnya, sebelum melakukan operasi penjumlahan, control unit akan memastikan bahwa data yang akan dijumlahkan sudah berada di register yang tepat.
- **Mengatur Waktu:** Control unit juga mengatur waktu pelaksanaan setiap operasi. Setiap operasi membutuhkan waktu tertentu untuk diselesaikan, dan control unit memastikan bahwa setiap operasi dimulai dan selesai pada waktu yang tepat.

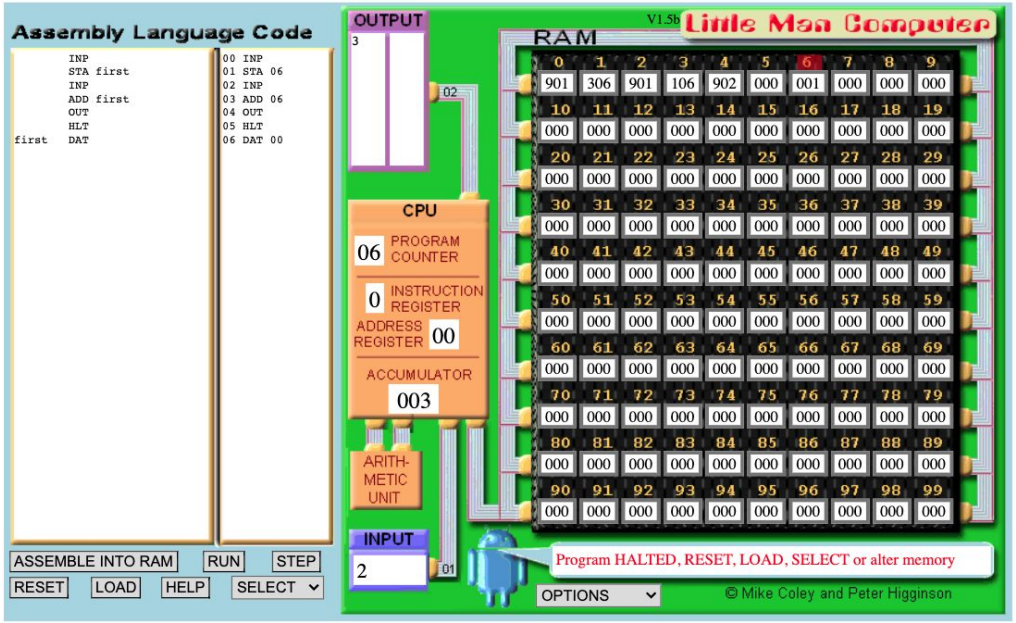
Control Unit

1. Menentukan sumber data (Source):
   - Register File
   - Memory
   - Immediate Value
   - ALU Result

2. Menentukan tujuan data (Destination):
   - Register Write
   - Memory Write
   - Program Counter Update



Praktik Menggunakan Little Man Computer (LMC)



Little Man Computer (LMC) adalah model simulasi komputer sederhana yang dirancang untuk tujuan pendidikan. Dikembangkan oleh Dr. Stuart Madnick di MIT pada awal 1960-an, LMC digunakan untuk mengajarkan konsep dasar arsitektur komputer dan pemrograman tingkat rendah kepada pemula. Sistem ini terdiri dari CPU sederhana (dengan Accumulator, Program Counter, dan Instruction Register), RAM terbatas, serta input dan output dasar. LMC menggunakan set instruksi yang sangat terbatas dan merepresentasikan data dalam bentuk angka desimal 3 digit.

LMC mendemonstrasikan prinsip-prinsip penting seperti siklus fetch-decode-execute, penyimpanan program, dan eksekusi sekuensial instruksi.

URL : https://www.peterhigginson.co.uk/lmc/

Praktik Menggunakan Little Man Computer (LMC)

- **Memahami Arsitektur von Neumann**: LMC memberikan representasi visual dan interaktif dari arsitektur von Neumann, membantu mahasiswa memahami bagaimana komponen utama (CPU, memori, unit I/O) berinteraksi dalam sistem komputer.
- **Mempelajari Siklus Instruksi**: Mahasiswa dapat mengamati dan memahami siklus fetch-decode-execute secara langsung, memberikan wawasan tentang bagaimana komputer memproses instruksi pada tingkat dasar.
- **Pengenalan Pemrograman Assembly**: LMC menggunakan bahasa assembly sederhana, memperkenalkan mahasiswa pada pemrograman tingkat rendah dan menjembatani pemahaman antara bahasa tingkat tinggi dan operasi mesin.
- **Mengeksplorasi Alur Data dan Kontrol**: Praktik dengan LMC memungkinkan mahasiswa melihat bagaimana data mengalir antar komponen dan bagaimana alur kontrol program dapat dimanipulasi, memberikan pemahaman praktis tentang konsep-konsep ini.
- **Pengembangan Pemikiran Logis dan Debugging**: Menulis dan men-debug program LMC membantu mahasiswa mengembangkan kemampuan berpikir logis, algoritmik, dan keterampilan pemecahan masalah yang penting dalam ilmu komputer.

Praktik Menggunakan Little Man Computer (LMC)

- INP (Input) :Menerima input dari pengguna dan menyimpannya di akumulator.
- OUT (Output) :Menampilkan nilai yang ada di akumulator.
- ADD (Add) : Menambahkan nilai dari alamat memori yang ditentukan ke akumulator.
- SUB (Subtract) : Mengurangi nilai dari alamat memori yang ditentukan dari akumulator.
- STA (Store) : Menyimpan nilai dari akumulator ke alamat memori yang ditentukan.
- MUL (Multiply) : Mengalikan nilai yang ada di accumulator dengan nilai dari alamat memory yang ditentukan
- LDA (Load) : Memuat nilai dari alamat memori yang ditentukan ke akumulator
- BRA (Branch Always) : Melompat ke alamat memori yang ditentukan tanpa syarat.
- BRZ (Branch if Zero) : Melompat ke alamat memori yang ditentukan jika nilai di akumulator adalah nol
- BRP (Branch if Positive) : Melompat ke alamat memori yang ditentukan jika nilai di akumulator adalah positif atau nol
- HLT atau COB (Halt) : Menghentikan eksekusi program
- DAT (Data) : Mendefinisikan nilai konstan atau menyediakan lokasi memory untuk menyimpan data

Praktik Menggunakan Little Man Computer (LMC)

Input/Output:

- 901: INP (Input) - Menerima input dari pengguna
- 902: OUT (Output) - Menampilkan nilai dari accumulator

Load/Store:

- 5xx: LDA (Load) - Memuat nilai dari alamat xx ke accumulator
- 3xx: STA (Store) - Menyimpan nilai accumulator ke alamat xx

Aritmetika:

- 1xx: ADD - Menambahkan nilai dari alamat xx ke accumulator
- 2xx: SUB - Mengurangkan nilai dari alamat xx dari accumulator

Branch/Jump:

- 6xx: BRA (Branch Always) - Melompat ke alamat xx
- 7xx: BRZ (Branch if Zero) - Melompat ke xx jika accumulator nol
- 8xx: BRP (Branch if Positive) - Melompat ke xx jika accumulator positif atau nol

Praktik Menggunakan Little Man Computer (LMC)

Lain-lain:

- 000: HLT (Halt) - Menghentikan program
- 0xx: DAT - Mendefinisikan konstanta atau variabel dengan nilai xx

Praktik Menggunakan Little Man Computer (LMC)

Format Penulisan Program Tiap Linenya:

xxx III yyy // xxx = alamat memory, III = instruksi, yyy = alamat operand

Contoh

```
000 LDA 010 // 000 = alamat memory, LDA = instruksi, 010 = alamat operand
001 ADD 011
002 OUT
003 HLT
010 DAT 5 // Menyimpan data
```

Praktik Menggunakan Little Man Computer (LMC)

Proses Penambahan 2 Bilangan

```
        INP
        STA first
        INP
        ADD first
        OUT
        HLT
first   DAT
```

Praktik Menggunakan Little Man Computer (LMC)

Praktik 1 : Jelaskan apa yang terjadi pada program berikut ini

Program 1

```
INP
STA 45
INP
ADD 67
OUT
HLT
```

Program 2

```
INP
STA 50
INP
ADD 50
OUT
HLT
```

Praktik Menggunakan Little Man Computer (LMC)

Tugas : Buatlah program LMC yang akan:

1. Membaca sebuah bilangan dari input dan simpan hasilnya di alamat memori 99
2. Menggandakan bilangan tersebut (buat jumlahnya 2 kali lipat dari jumlah nilai yang diinput)
3. Menampilkan hasil penggandaan ke output.