STIKOM BALI

ORGANISASI KOMPUTER

Materi 6: Control Unit Operations

I Nyoman Kusuma Wardana

Sistem Informasi STMIK STIKOM Bali

Materi Perkuliahan

- **Pendahuluan**
- **Siklus Instruksi**
- **Micro-operations**

STIKOM BALI

PENDAHULUAN

PENDAHULUAN

- Jika kita tahu:
  - **set instruksi mesin** (meliputi efek setiap ***opcode*** & pmahaman **mode pngalamatn**)
  - **register**[2] yg terlibat
- **Maka** → kita akan dpt memahami apa **fungsi** yg hrs dilakukan oleh **prosesor**.

- Ini saja **blm lengkap**, kita jg hrs paham → **antarmuka eksternal** & bgmn **interupsi** di-*handle*

PENDAHULUAN

- Komponen2 brikut yg **mnentukan spesifikasi** dr **prosesor**:

1. **Operations (opcode)**
2. **Addressing Modes**
3. **Register**

Didefinisikan oleh set instruksi

4. **Modul I/O**
5. **Modul Memori**

Didefinisikan umumnya oleh sistem bus

6. **Interupsi**

Didefinisikan sebagian oleh bus & sebagian oleh fitur prosesor

PENDAHULUAN

- **Pertanyaan** kita skrg:
- Bgmn fungsi2 ini dpt dijalankan? Atau…
- Bgmn brbagai elemen ini dpt dikontrol utk menjalani fungsinya?

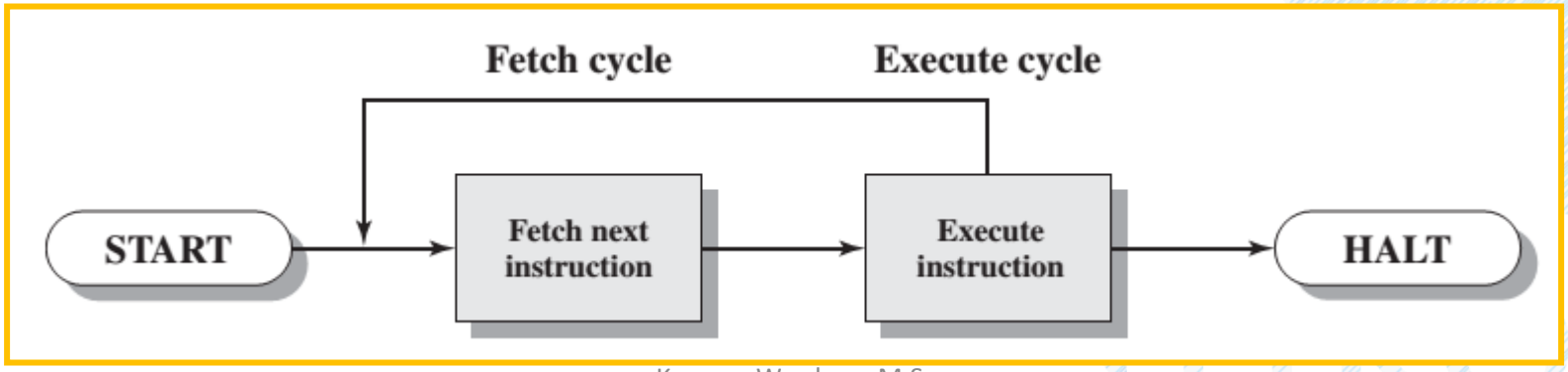
- Dgn demikian, pembahasan kita selanjutnya adlh tentang → **Control Unit**
- **Control Unit** → sistem yg **mengontrol kinerja Prosesor**

STIKOM BALI

SIKLUS INSTRUKSI

SIKLUS INSTRUKSI

- Eksekusi program → proses perulangan ***instruction fetch*** dan ***instruction execution***
- Proses yg diperlukan dlm sekali instruksi dikenal sebagai **siklus instruksi** (***instruction cycle***)
- ***Instruction cycle*** trdr dr : ***fetch cycle*** dan ***execute cycle***



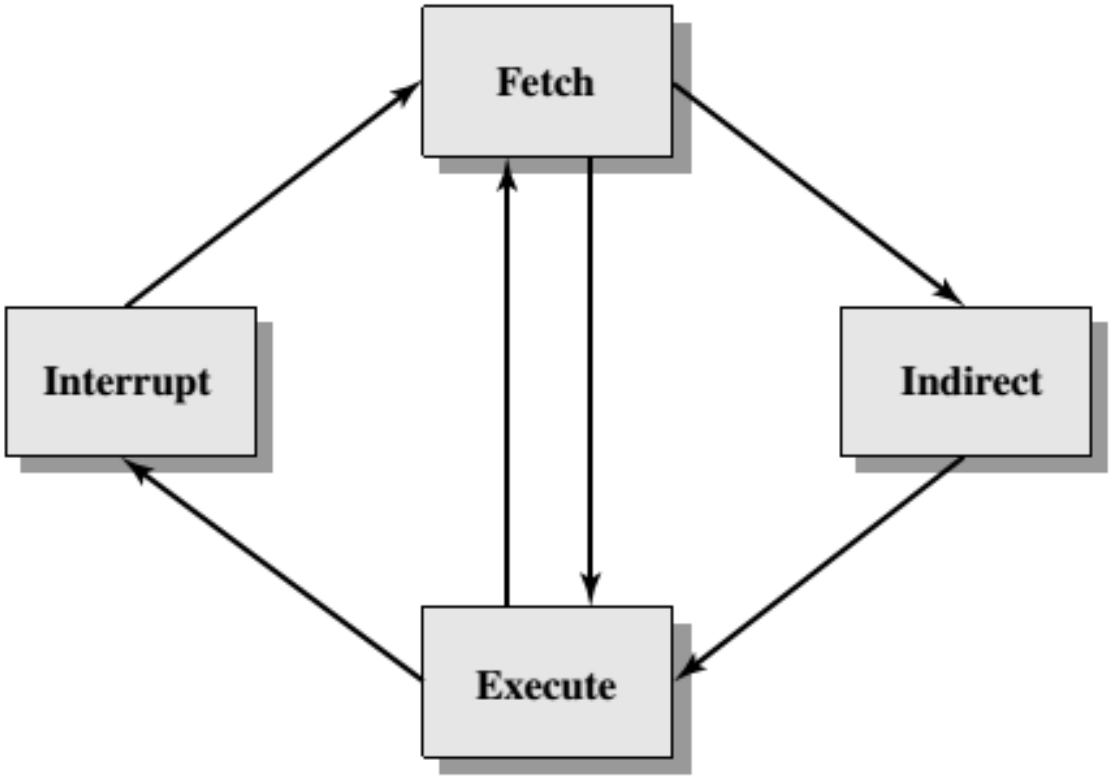
Siklus Instruksi

- Siklus instruksi dpt trdr dr **tahapan²** berikut:

1. ***Fetch*** → **membaca** instruksi berikutnya dr memori ke prosesor
2. ***Execute*** → **menterjemahkan** opcode & melaksanakan operasi yg ditentukan
3. ***Interrupt*** → jika layanan interupsi diaktifkan & terjd interupsi, maka **simpan proses** saat ini dan **layani interupsi** tsb
4. Mungkin trjd ***indirect*** → memerlukan **alamat** & **tahapan tambahan**

Siklus Instruksi

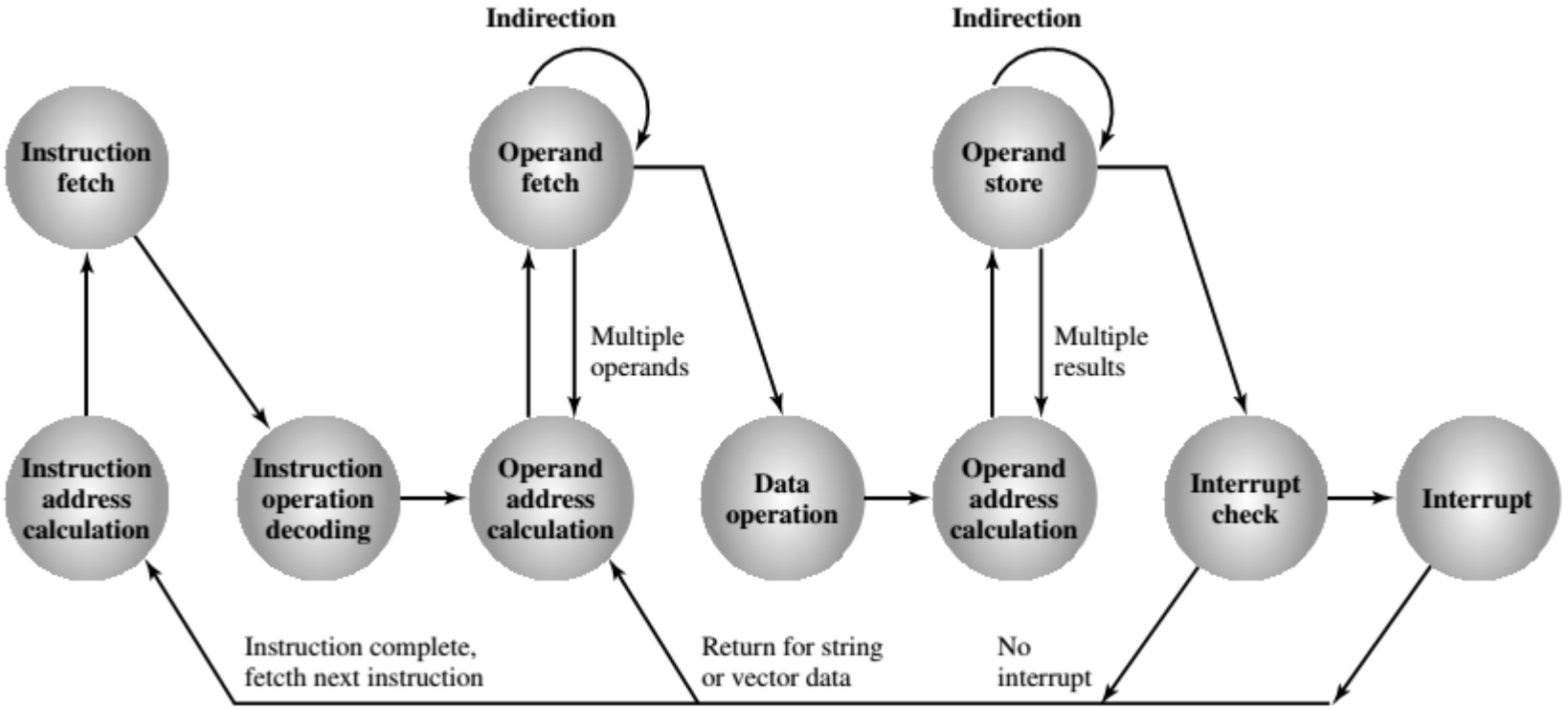
- Amati ilustrasi siklus intruksi sbb:



Siklus Intsruksi

Siklus Instruksi

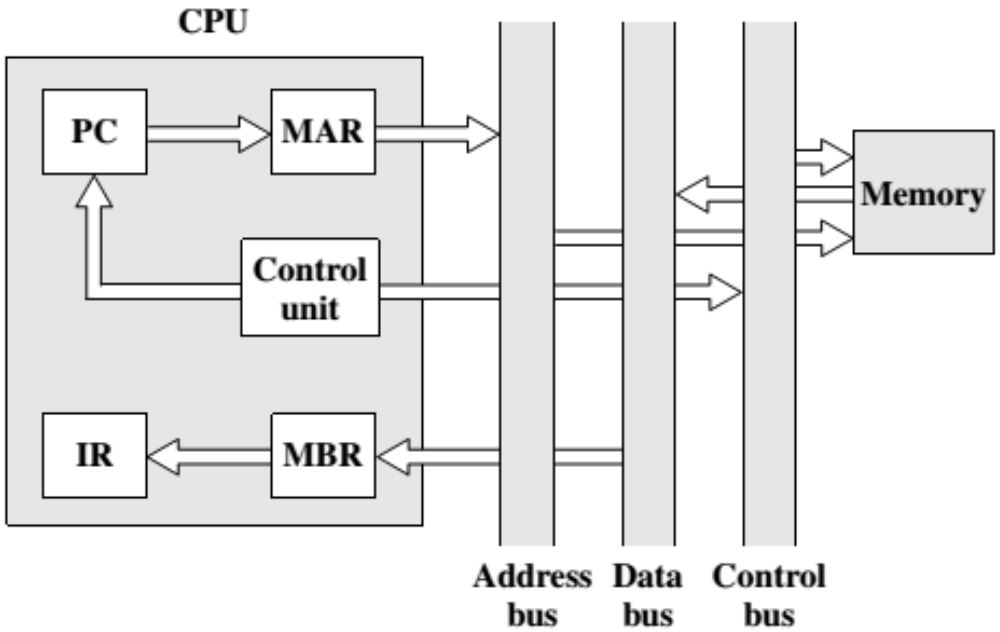
- Amati ilustrasi siklus intruksi yg lebih lengkap sbb:



Instruction Cycle State Diagram

SIKLUS INSTRUKSI

- Amati gb. berikut:



MBR = Memory buffer register
MAR = Memory address register
IR = Instruction register
PC = Program counter

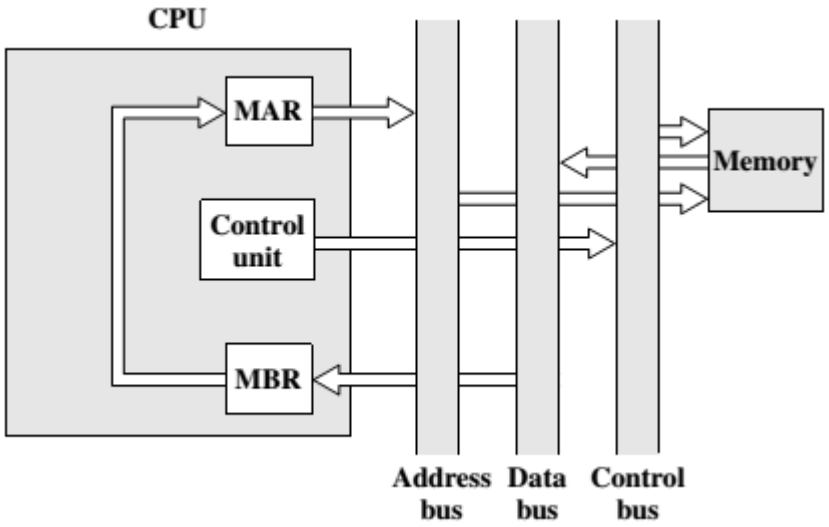
Data Flow, Fetch Cycle

SIKLUS INSTRUKSI

Penjelasan:

- **PC** berisikan alamat utk instruksi selanjutnya. Alamat ini akan dipindahkan ke **MAR** & selanjutnya ditempatkan pd **address bus**
- **Control Unit (CU)** meminta utk membaca isi memori & hasilnya akan ditempatkan di **data bus**. Hasil pd data bus ini akan disalin ke **MBR** dan selanjutnya dipindahkan ke IR
- Ketika siklus fetch selesai, maka CU akan mengevaluasi isi **IR**. Jika ada ***indirect***, maka lakukan operasi indirect tsb.

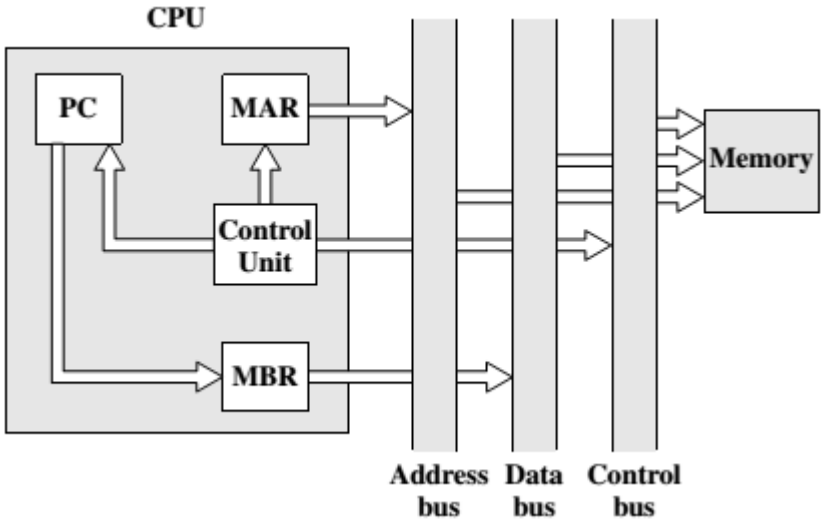
SIKLUS INSTRUKSI



Data Flow, Indirect Cycle

- Jika trjd ***operasi indirect*** → maka alamat referensi di **MBR** akan disalin ke **MAR**.
- Selanjutnya, CU akan **meminta** pembacaan memori utk mndapatkan alamat operand yg diinginkan
- Alamat ini yg akan disalin ke **MBR**

Siklus Instruksi



Data Flow, Interrupt Cycle

- Bgmn jk terjd **interupsi**?
- Isi terakhir dr **PC** hrs disimpan sehingga dpt di-resume setelah interupsi
- Alamat dr **PC** akan ditulis ke memori melalui **MBR**
- Dr **CU**, alamat di memori khusus (misal *stack pointer*) akan disalin ke **MAR**
- Selanjutnya, **PC** akan dimuati dgn alamat2 dr operasi interupsi

STIKOM BALI

MICRO-OPERATIONS

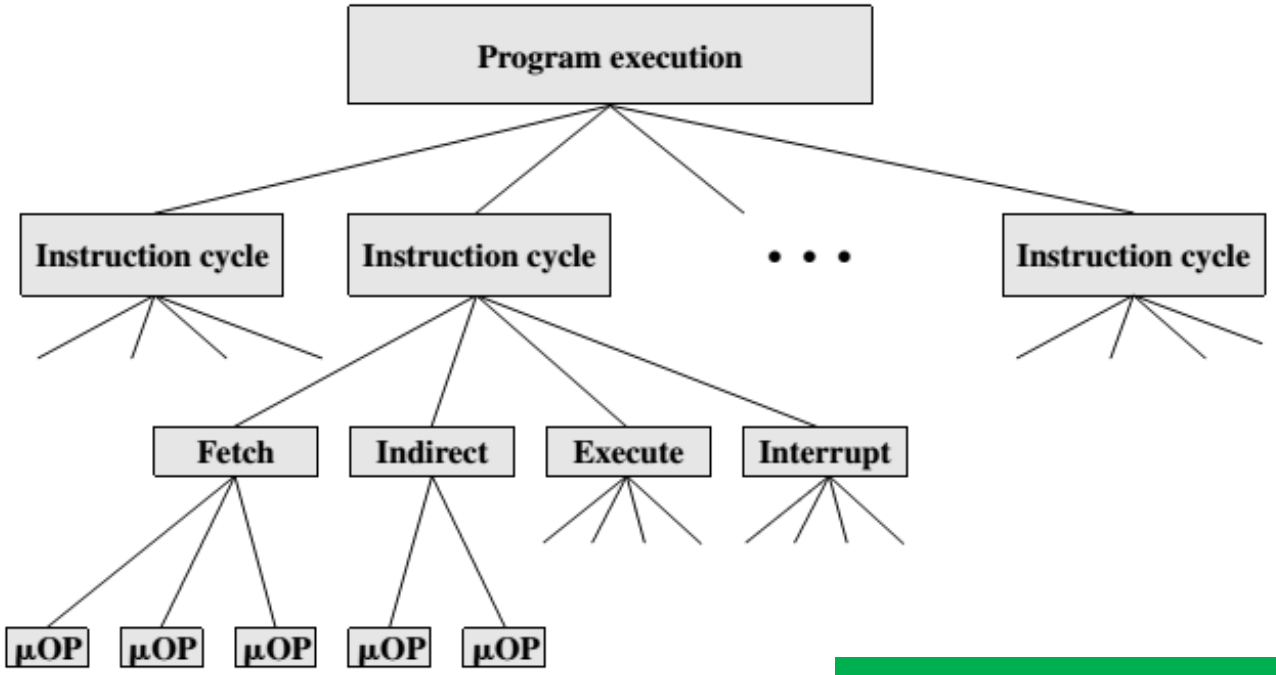
Micro-Operations

- **Setiap instruksi** → mungkin jg **terdiri dr bbrp unit2 (siklus) yg lebih kecil**
- Setiap siklus yg lebih kecil melibatkan **rangkaian langkah2**, dimana tiap langkah tsb akan **melibatkan register**
- **Langkah2 kecil** ini dikenal sbg:

  micro-operations

Micro-Operations

- ***Micro*** → setiap langkahnya sederhana
- Op. mikro → operasi "***atomic***" dr prosesor



Elemen[2] pokok dr eksekusi program

Micro-Operations

Fetch Cycle

- ***Fetch cycle*** → trjd **diawal eksekusi** program dgn **membaca** instruksi dr memori

- Asumsi trdpt **4 register** yg terlibat:
  1. ***Memory Address Reg***. (**MAR**)
  2. ***Memory Buffer Reg***. (**MBR**)
  3. ***Program Counter*** (**PC**)
  4. ***Instruction Reg***. (**IR**)

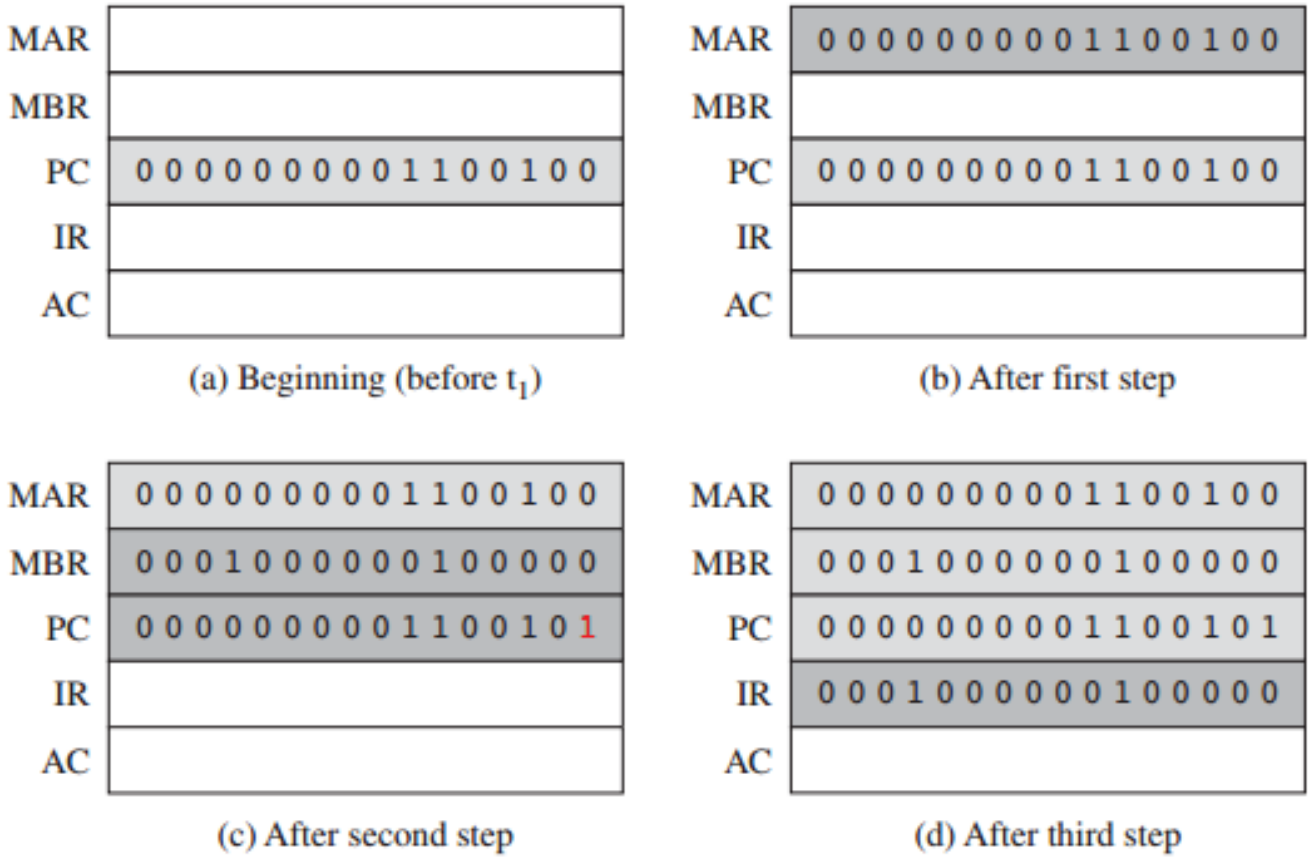
Micro-Operations

Penjelasan:

- **MAR** → terhubung dgn ***address bus*** utk menentukan **alamat** di memori utk operasi baca atau tulis
- **MBR** → terhubung dgn ***data bus*** yg berisikan **nilai** (data) utk **ditulis** ke memori, atau **nilai** terakshir yg **dibaca** dr memori
- **PC** → menyimpan alamat utk instruksi selanjutnya
- **IR** → menyimpan instruksi terakhir yg terbaca

Micro-Operations

- Amati perubahan isi register dlm prosesor, sbb:



Urutan event, fetch cycle

Micro-Operations

Penjelasan:

- **Pada awal** siklus pembacaan (***fetch cycle***), alamat dr instruksi berikutnya akan tersimpan di **PC**, **contoh**: **1100100**
- **Langkah pertama** → pindahkan alamat pd **PC** ke **MAR**. Hal ini dikarenakan hanya register **MAR** yg terhubung dgn **address bus**
- **Langkah kedua** → baca di **memori** alamat ini dgn melakukan operasi READ pd **control bus**. Hasilnya disimpan pd **MBR**. Disaat bersamaan, naikkan **PC** dgn 1 (**PC = PC+1**)
- **Langkah ketiga** → pindahkan **MBR** ke **IR**

Micro-Operations

- Amati bahwa sebuah ***siklus fetch*** sederhana terdiri dr: **3 langkah** dan **4 micro-operations**
- Setiap ***micro-operation*** melibatkan **perpindahan data pd register**
- Selama perpindahan ini **tidak saling terkait** (misal membaca dr memori dgn menaikkan nilai PC) → maka kedua operasi dpt dilakukan dlm **1 langkah** (hemat waktu)

Micro-Operations

- Secara simbolis, urutan event dpt ditulis sbb:

t_1: MAR ← (PC)
t_2: MBR ← Memory
PC ← (PC) + I
t_3: IR ← (MBR)

- Dgn ***I*** → panjang instruksi (bisa 1, 2,dsb)
- ***t1***, ***t2***, ***t3*** → time unit

Micro-Operations

t_1: MAR ← (PC)

t_2: MBR ← Memory

PC ← (PC) + *I*

t_3: IR ← (MBR)

- **Unit waktu I**: salin isi ***PC*** ke ***MAR***
- **Unit waktu II**: salin isi memori yg ditentukan oleh ***MAR*** ke ***MBR***. Naikkan ***PC*** dgn ***I***
- **Unit waktu III**: salin isi ***MBR*** ke ***IR***

Micro-Operations

- Pengelompokkan jg dpt ***disusun ulang***, sbb:

t_1: MAR ← (PC)
t_2: MBR ← Memory
t_3: PC ← (PC) + I
IR ← (MBR)

- Operasi mikro **ketiga** dan **keempat** dpt digabungkan pd ***t3***

Micro-Operations

- Pengelompokan operasi mikro harus mengikuti **2 aturan sederhana**, sbb:

1. **Urutan event** yg tepat hrs dipenuhi

   Misal, pd contoh sblmnya: (***MAR ← (PC)***) HARUS dilakukan sblm (***MBR ← Memory***)

2. **Konflik** hrs dihindari

   Hindari membaca & menulis pd register yg sama pd saat yg sama.

   Contoh: (***MBR ← Memory***) dan (***IR ←MBR***) hrs dihindari dilakukan pd saat yg sama

Daftar Pustaka

- Abd-El-Barr, M., El-Rewini, H., ***Fundamentals of Computer Organization and Architecture***, John Wiley&Sons, Inc.
- Stallings, W., 2010, ***Computer Organization and Architecture: Designing for Performance*** 8th edition, Prentice Hall
- Hamacher,C., Vranezic, Z., Zaky, S., Manjikian, N., 2012, ***Computer Organization and Embedded Systems*** 6th edition, McGrawHill