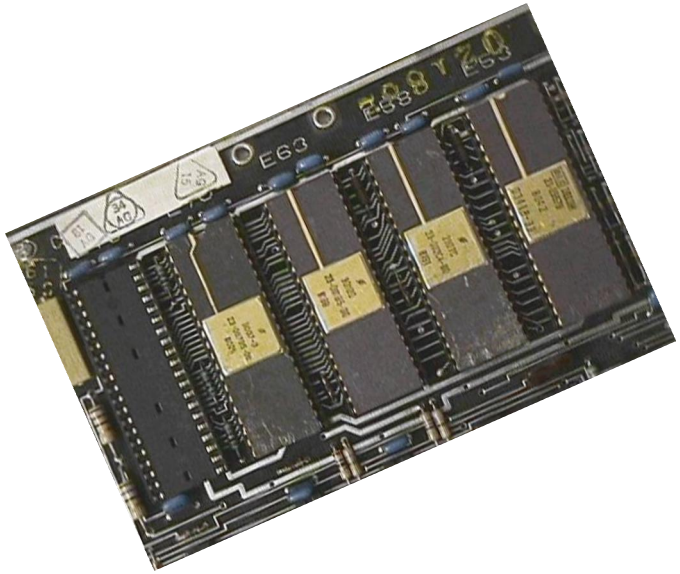
Tietokoneen rakenne

Ohjaus-yksikkö



Ch 16-17 [Sta06]

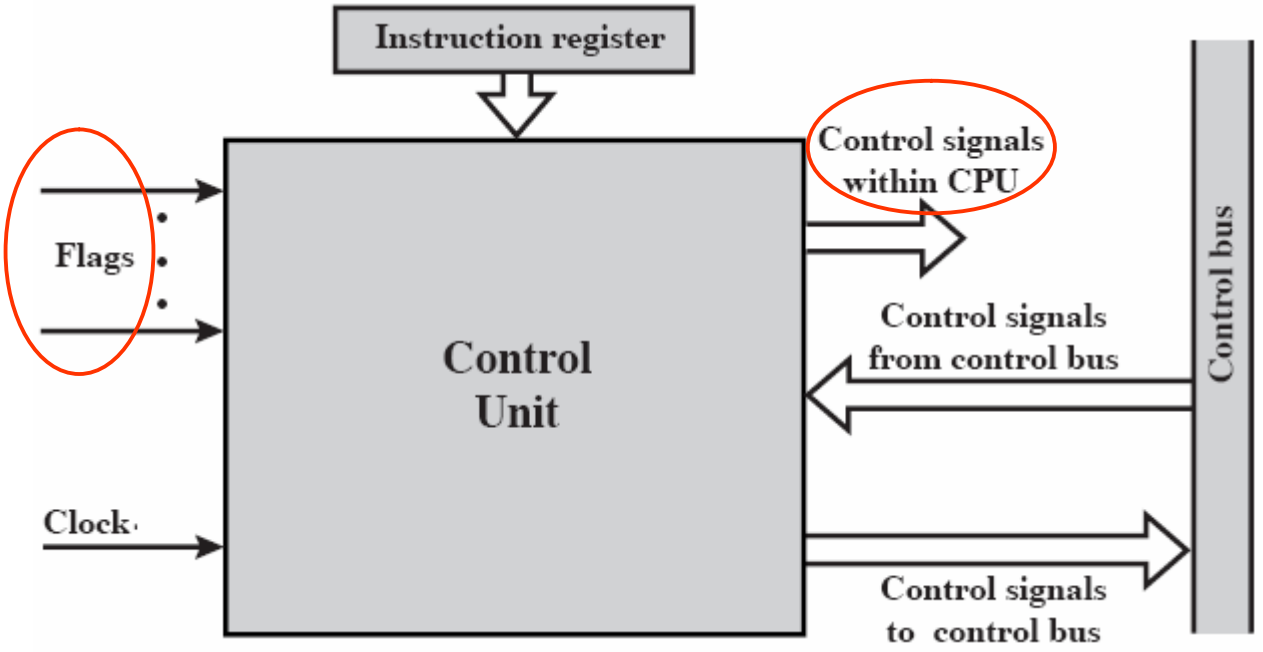
- Mikro-operaatiot
- Ohjaussignaalit
- Langoitettu ohjaus
- Mikro-ohjelmoitu ohjaus

Mitä ohjaus/kontrolli tarkoittaa?

- **Arkkitehtuuri määrää CPU:n ulkoisen, ohjelmoijalle (myös KJ) näkyvän toiminnan**
  - Millainen käskykanta käytössä, mitä käskyt tekevät?
  - Mikä operaatio, missä operandit?
  - Miten keskeytykset hoidellaan?
- **CU määrää kuinka käskyssä kuvatut asiat saadaan tehdyiksi laitteistossa (CPU, MEM, väylät, I/O)**
  - Selvittää miten piirien täytyy toimia tietyllä hetkellä
  - Valitsee millaisia ohjaussignaaleja pitää antaa: mille piireille, milloin, missä järjestyksessä, ...
  - Fyysiset johtimet välittävät ohjaussignaalit piireille täsmällisesti kellopulssin ajoittamina

Ohjaussignaalit

(Sta06 Fig 16.4)



- **Ohjaus = signaalien / bittien siirtämistä paikasta toiseen**
  - CPU:n sisällä: REG ó REG, ALU ó REG, ALU-oper
  - CPU ó MEM (I/O-ohjain): osoite, data, ohjaus
- **Ajoitus, järjestys**

Mikro-operaatiot

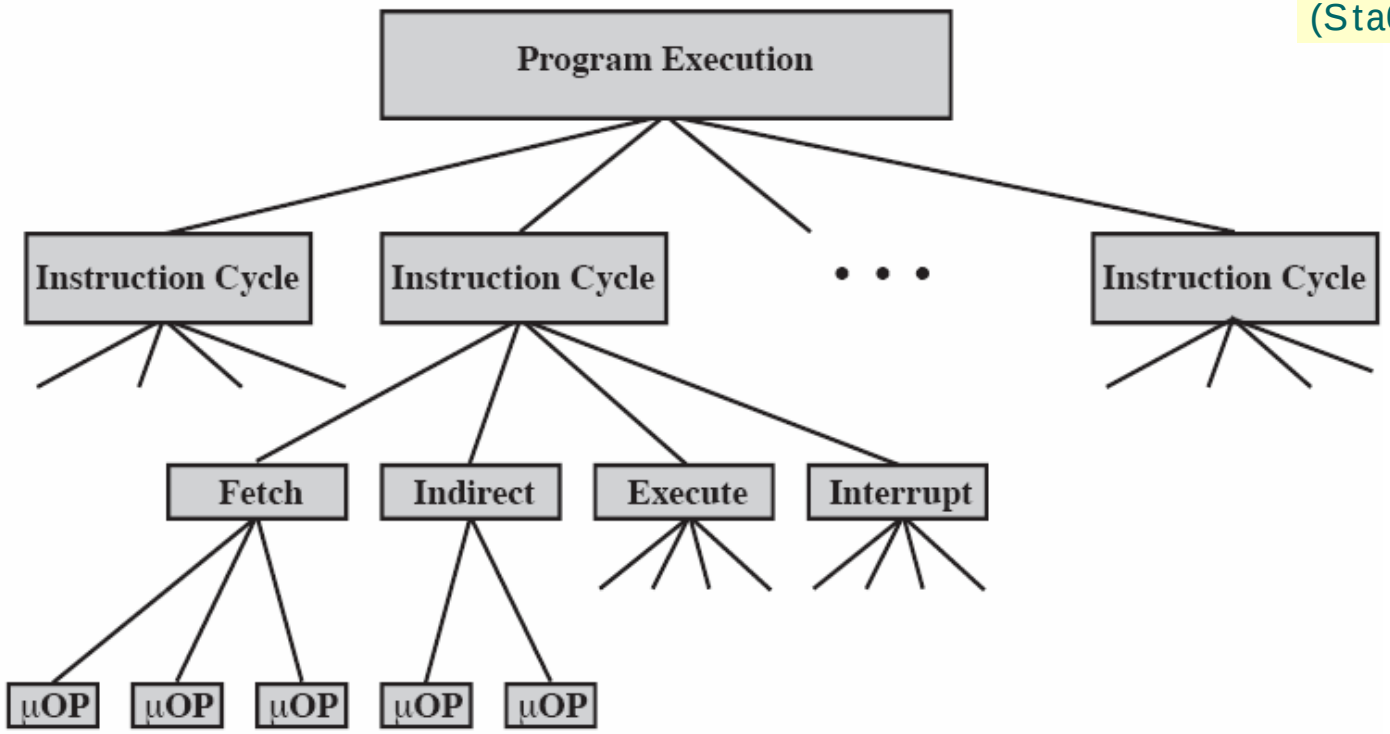
- **Yksinkertaisia ohjaussignaaleja, jotka aiheuttavat yhden pienen toiminnon (operaation)**
  - Esim. bitit siirtyvät rekisteristä r1 väylää pitkin edelleen ALU:un
- **Kellosyklin kesto pisimmän operaation perusteella**
- **Kunkin syklin aikana useita mikro-operaatioita**
  - Osa voidaan suorittaa samanaikaisesti, jos tapahtuvat eri osissa piirejä
  - Vältettävä resurssikonfliktit
    - WaR tai RaW, ALU, väylä
  - Osa suoritettava peräkkäin tietyssä järjestyksessä, semantiikan säilyttävä

t1: MAR ← PC
t2: MBR ← MEM[MAR]
PC ← PC + 1

Toteutus siten, että ei tarvita ALUa

Käskysykli

(Sta06 Fig 16.1)



- Kun mikro-operaatiot kohdistuvat laitteiston eri piireihin,

  laitteisto voi suorittaa useita samanaikaisesti
- Ks. luvun 12 käskysykliesimerkit

Käskyn noutosykli

Esim:

t1: MAR • PC

t2: MAR • MMU(MAR)

Control Bus • Reserve

t3: Control Bus • Read

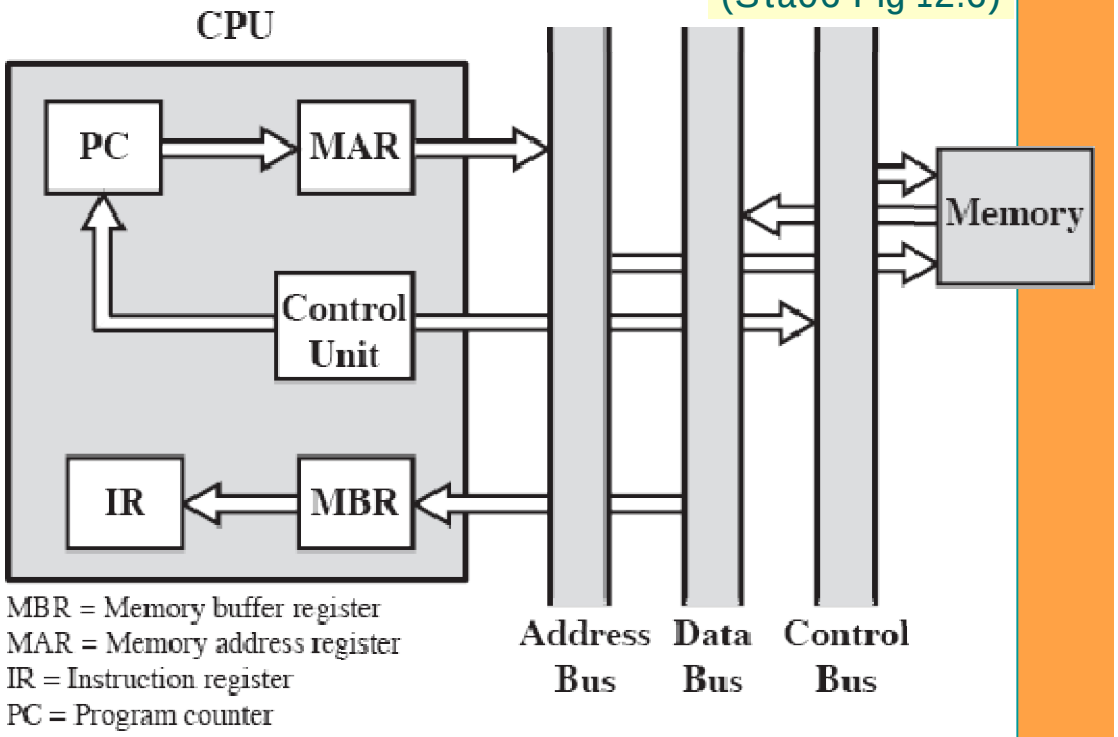
PC • PC + 1

t4: MBR • MEM[MAR]

Control Bus • Release

t5: IR • MBR

(Sta06 Fig 12.6)



Onko suoritusjärjestyksellä merkitystä?

Mitä voi suorittaa rinnakkain?

Mitä samaan sykliin, mitkä tarvitsevat oman syklin?

Käskysykli

- **Operandien noutosyklit**
  - Rekistereistä tai muistista
  - Osoitelaskenta
- **Suoritussyklit**
  - Suoritus tavallisesti ALUssa
  - Operandit sisäänmenoihin, ja ohjaus operaatiosta
  - Tulos ALU:n ulostulosta rekisteriin/muistiin
  - flags • status
- **Keskeytyssyklit**
  - ks. luvun 12 esimerkit: Pentium, PowerPC
  - Mitkä samaan mikrokäskyyn?
  - Mitkä samaan aikaan/peräkkäin?

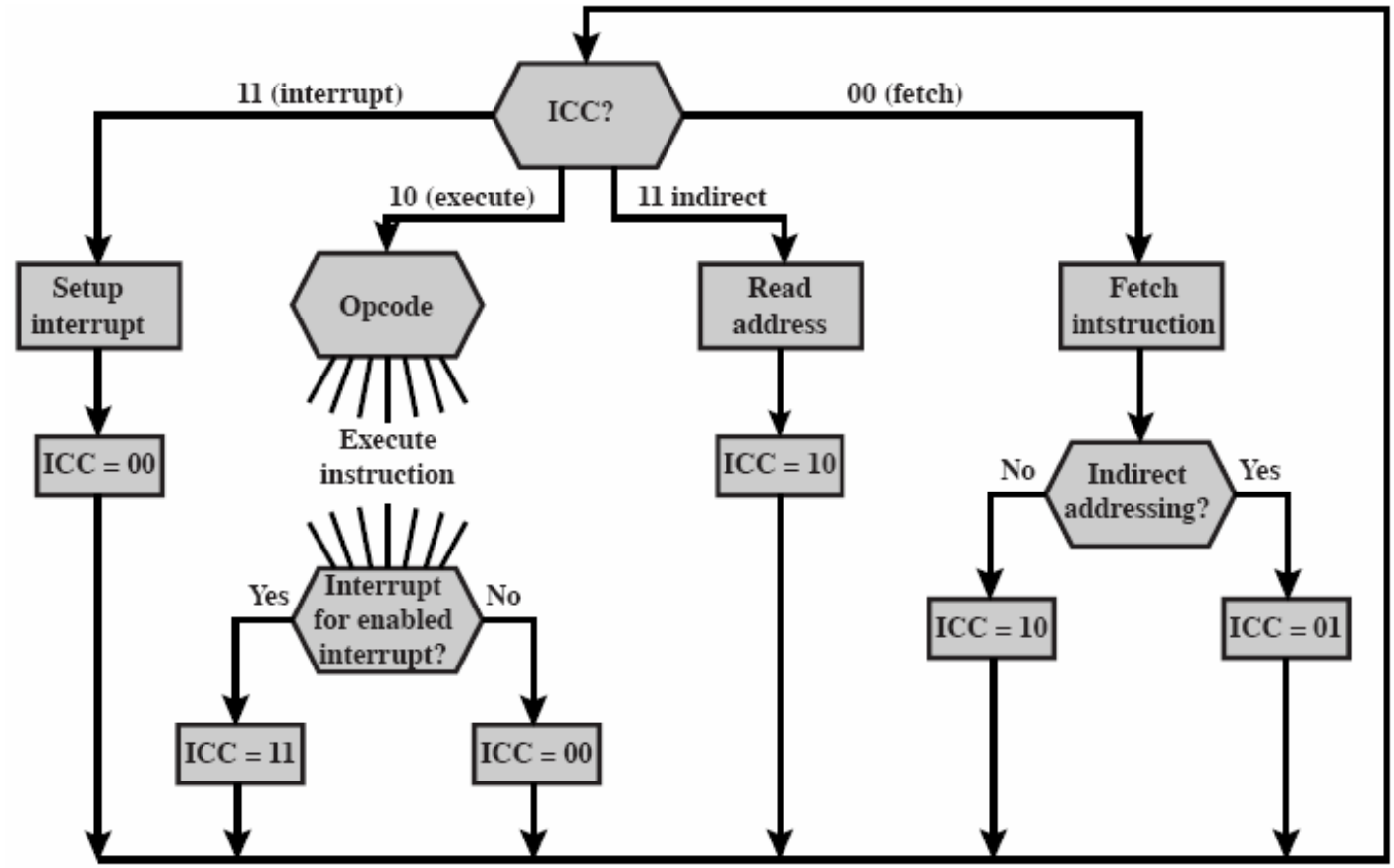
ADD r1,r2,r3:
t1: ALUin1 ← r2
t2: ALUin2 ← r3
ALUoper ← IR.oper
t3: r1 ← ALUout
flags ← xxx

ISZ X, Increment and Skip if zero:
t1: MAR ← IR.address
t2: MBR ← MEM[MAR]
t3: MBR ← MBR+1
t4: MEM[MAR] ← MBR
if (MBR=0) then PC ← PC +1

ehdollisuuskin onnistuu

Käskysykli tila-automaattina

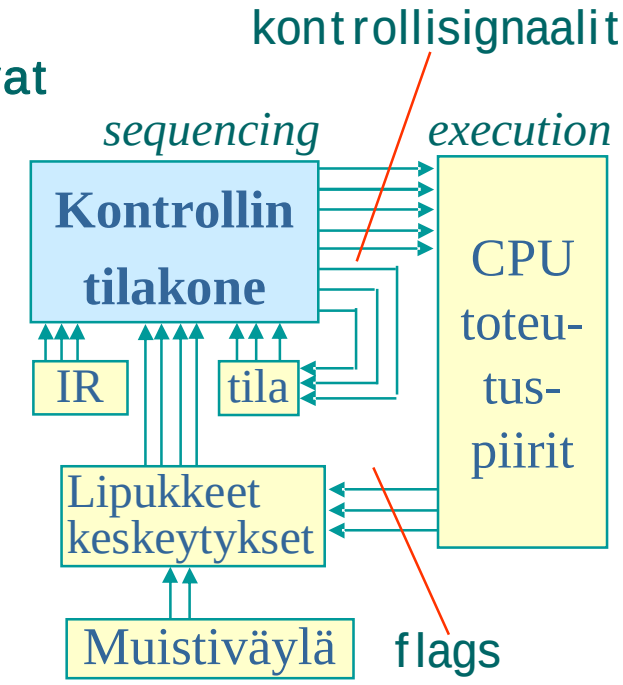
- ICC: Instruction Cycle Code register



(Sta06 Fig 16.3)

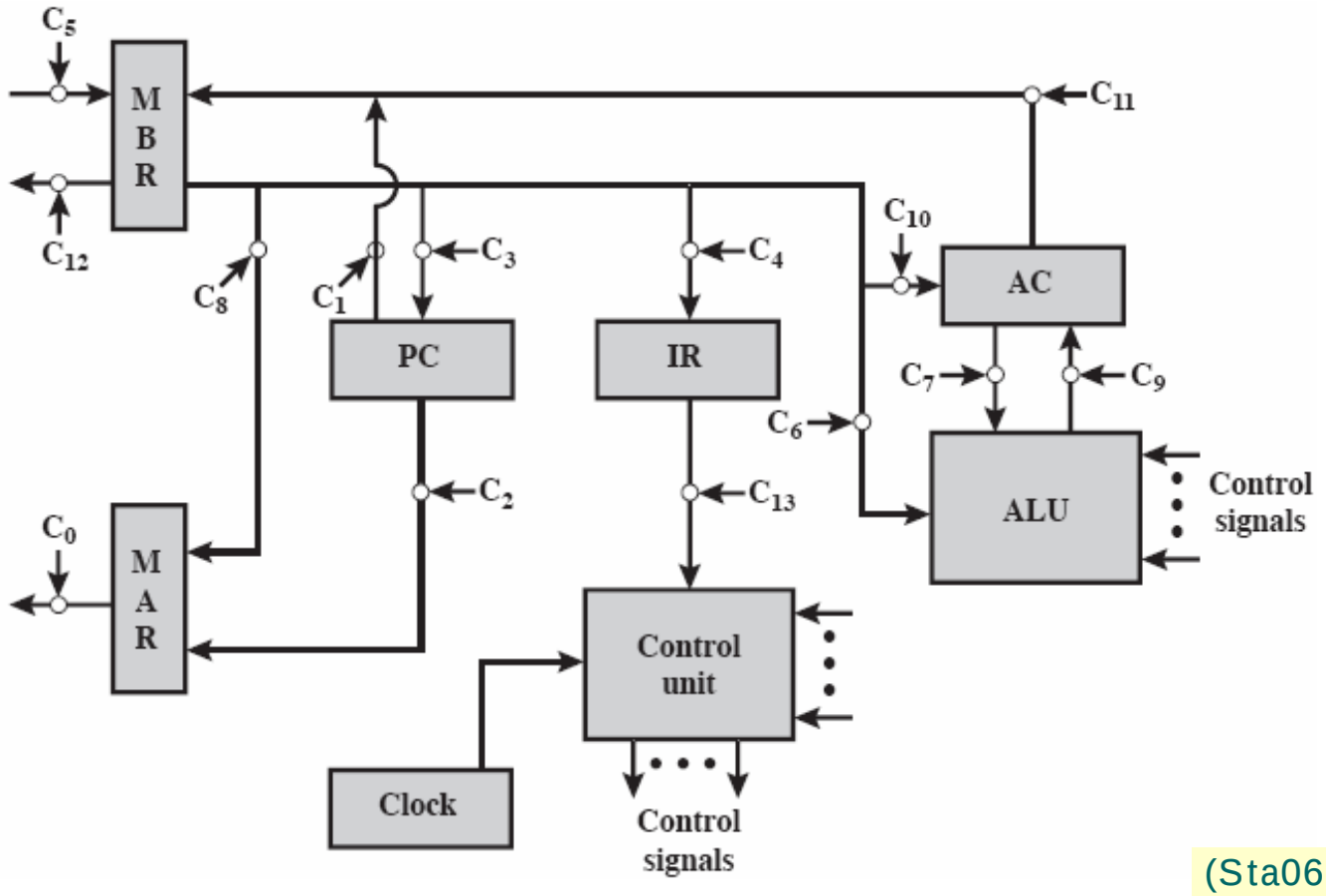
Käskysykli kontrolli tila-automaattina

- **CU:n toiminnan voi esittää tila-automaattina**
  - Tila: missä käskysyklin vaiheessa CPU menossa
  - Alitila: riippuu ajoituksesta, muodostuu ryhmästä mikro-operaatioita, jotka voi suorittaa yhdellä syklillä
- **Alitilan tuottamat uudet ohjaukset riippuvat**
  - Tilasta itsestään
  - IR-rekisterin kentistä (operaatio, osoittaminen)
  - Edellisistä tuloksista (flags)
    = Execution
- **Uusi tila edellisen tilan ja lipukkeiden perusteella**
  - Myös CPU:n ulkopuoliset keskeytykset vaikuttavat tilaan
    = Sequencing



Ohjaussignaalien tuottaminen

- Mikro-operaatio ð aktivoitava useita ohjaussignaaleja
- Esim: Yhden akkurekisterin arkkitehtuuri



(Sta06 Fig 16.5)

Ohjaussignaalien tuottaminen

Sta06 Fig 16.5

Micro-operations	Timing	Active Control Signals
Fetch:	t_1: MAR ← (PC)	C_2
	t_2: MBR ← Memory PC ← (PC) + 1	C_5, C_R ??
	t_3: IR ← (MBR)	C_4
Indirect:	t_1: MAR ← (IR(Address))	C_8
	t_2: MBR ← Memory	C_5, C_R
	t_3: IR(Address) ← (MBR(Address))	C_4
Interrupt:	t_1: MBR ← (PC)	C_1
	t_2: MAR ← Save-address PC ← Routine-address	??
	t_3: Memory ← (MBR)	C_{12}, C_W

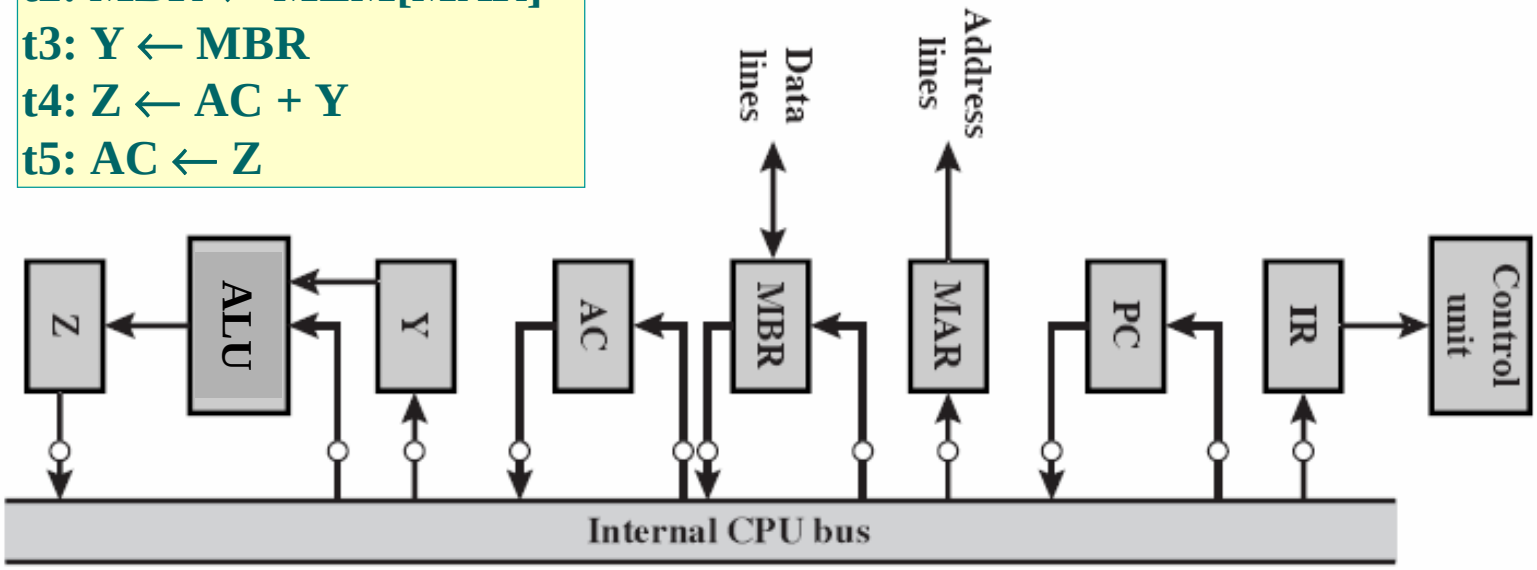
C_R = Read control signal to system bus.
C_W = Write control signal to system bus.

(Sta06 Table 16.1)

Yleinen organisointi

- Kuvan 16.5 organisaatio monimutkainen toteutettavaksi
- Komponentit tavallisesti yhteisen väylän varteen
- ALU:lle apurekisterit: X ja Y

ADD I:
t1: MAR ← IR.address
t2: MBR ← MEM[MAR]
t3: Y ← MBR
t4: Z ← AC + Y
t5: AC ← Z



(Sta06 Fig 16.6)

Tietokoneen rakenne

Langoitettu ohjaus

Langoitettu ohjausyksikkö (hardwired)

- Kun tiedossa CU:n sisäänmenot ja ulostulot
  - Toiminnallisuus kuvattavissa boolen logiikalla
  - Ohjausyksiköstä voi muodostaa yhden loogiikkapiirin
- Esim. $C_5 = \overline{P}*\overline{Q}*T_2 + P*\overline{Q}*(LDA)*T_2 + \ldots$

Fig 16.3, 16.5 ja Tbl 16.1

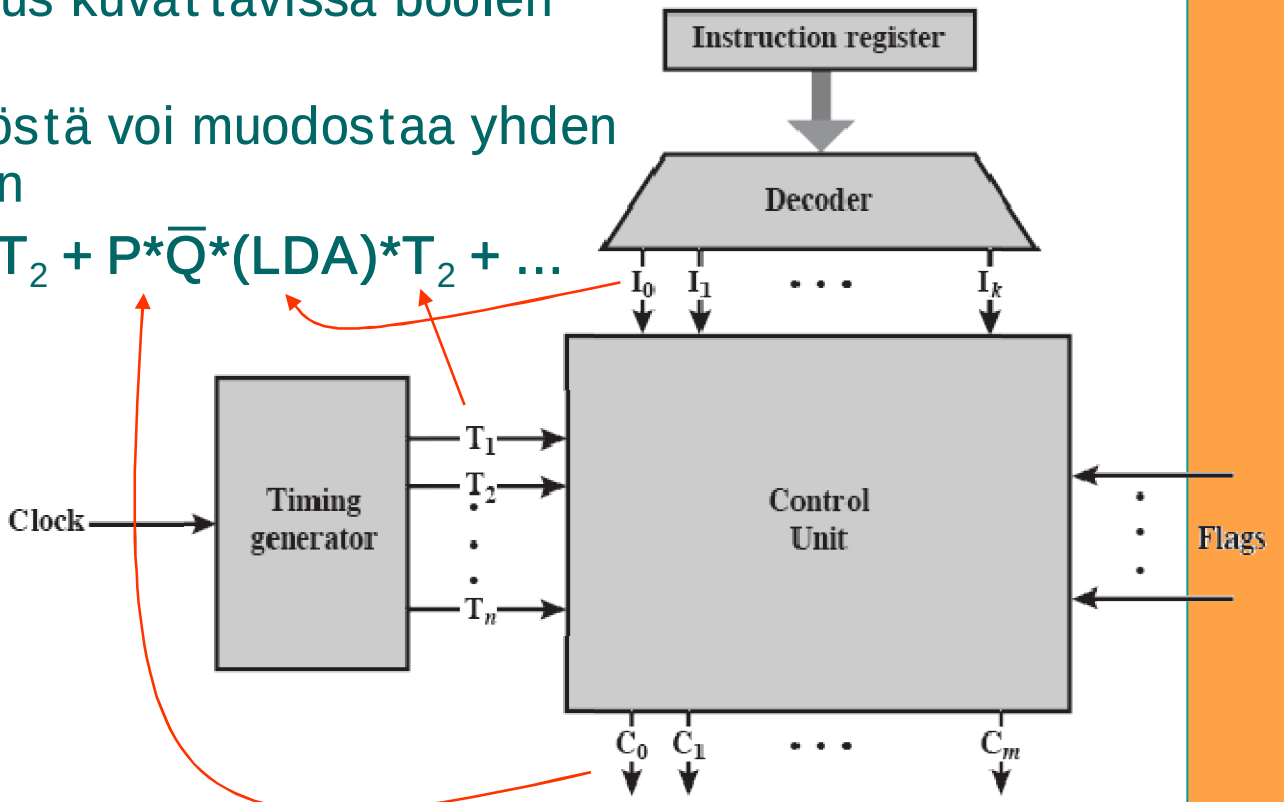
ICC:lle bitit P ja Q

PQ = 00 Fetch Cycle

PQ = 01 Indirect Cycle

PQ = 10 Execute Cycle

PQ = 11 Interrupt Cycle



(Sta06 Fig 16.10)

Langoitettu ohjausyksikkö

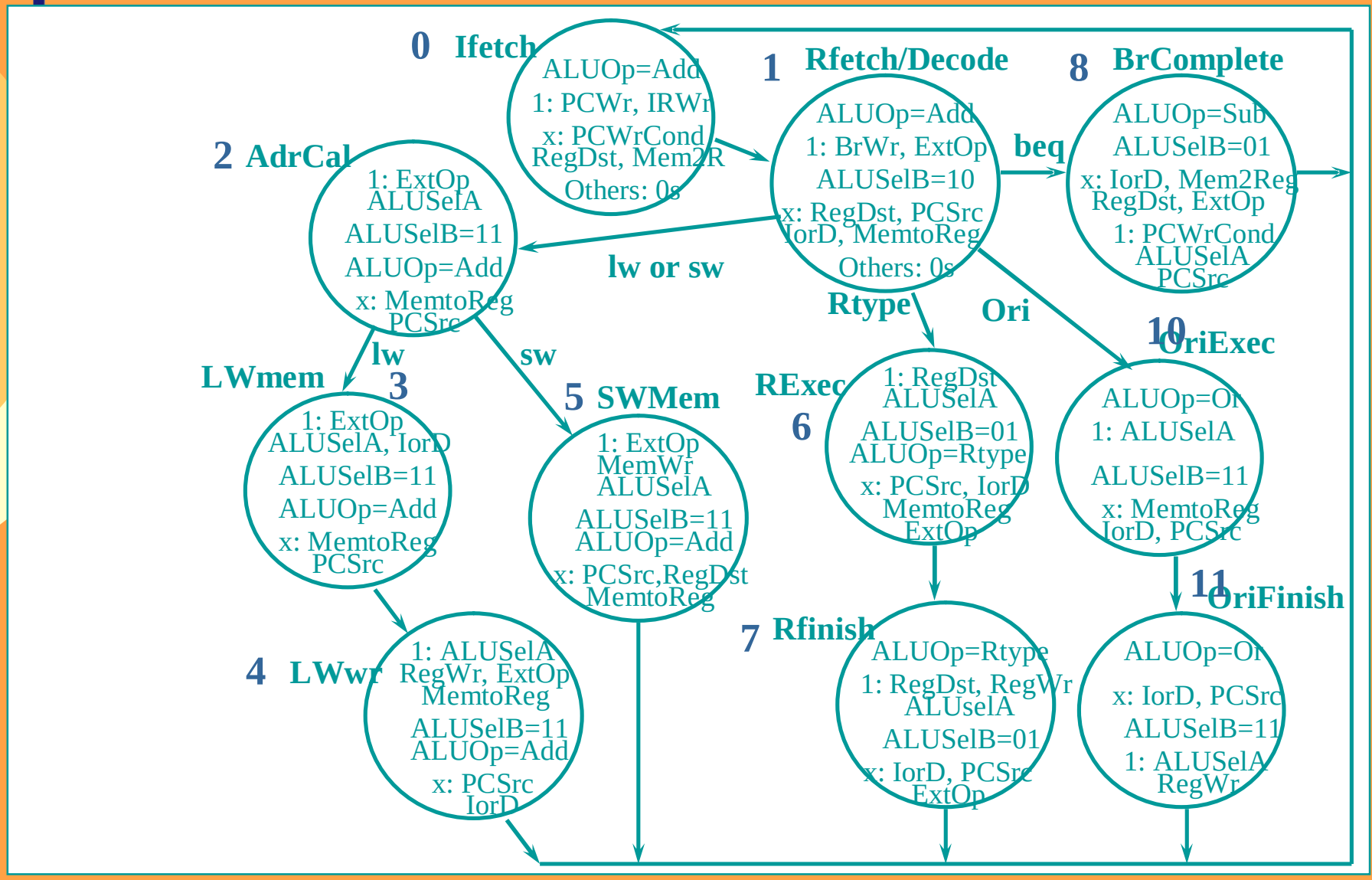
- **Decoder (4-to-16)**
  - 4:n bitin käskykoodista yksikäsitteinen ohjaus CU:lle
  - Vain yksi signaali kerrallaan aktiivisena

I1	I2	I3	I4	O1	O2	O3	O4	O5	O6	O7	O8	O9	O10	O11	O12	O13	O14	O15	O16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Esim: opcode = 5 (bitit I1, I2, I3, I4) ⟶ signaali O11 on tosi (1)

(Sta06 Table 16.3)

Äärellinen tila-automaatti



Tilasiirtymät

Next state from current state

- State 0 -> State1
- State 1 -> S2, S6, S8, S10
- State 2 -> S5 or ...
- State 3 -> S9 or ...
- State 4 ->State 0
- State 5 -> State 0
- State 6 -> State 7
- State 7 -> State 0
- State 8 -> State 0
- State 9-> State 0
- State 10 -> State 11
- State 11 -> State 0

Alternatively,
prior state & condition

S4, S5, S7, S8, S9, S11 -> State0

_____ -> State1

_____ -> State 2

_____ -> State 3

_____ -> State 4

State2 & op = SW -> State 5

_____ -> State 6

State 6 -> State 7

_____ -> State 8

State3 & op = JMP -> State 9

_____ -> State 10

State 10 -> State 11

Langoitettu ohjaus

- **Ohjaussignaalien generointi suoraan laitteistolla nopeaa**
- **Heikkouksia**
  - CU vaikea suunnitella
    - Piiristä tulee helposti suuri ja monimutkainen
  - CU vaikea muuttaa
    - Suunnittelu ja piirin ”minimointi” uusiksi
- **RISC-filosofia helpottaa**
  - Yksinkertainen käskykanta ja toteutus

Mikro-ohjelmoitu ohjaus

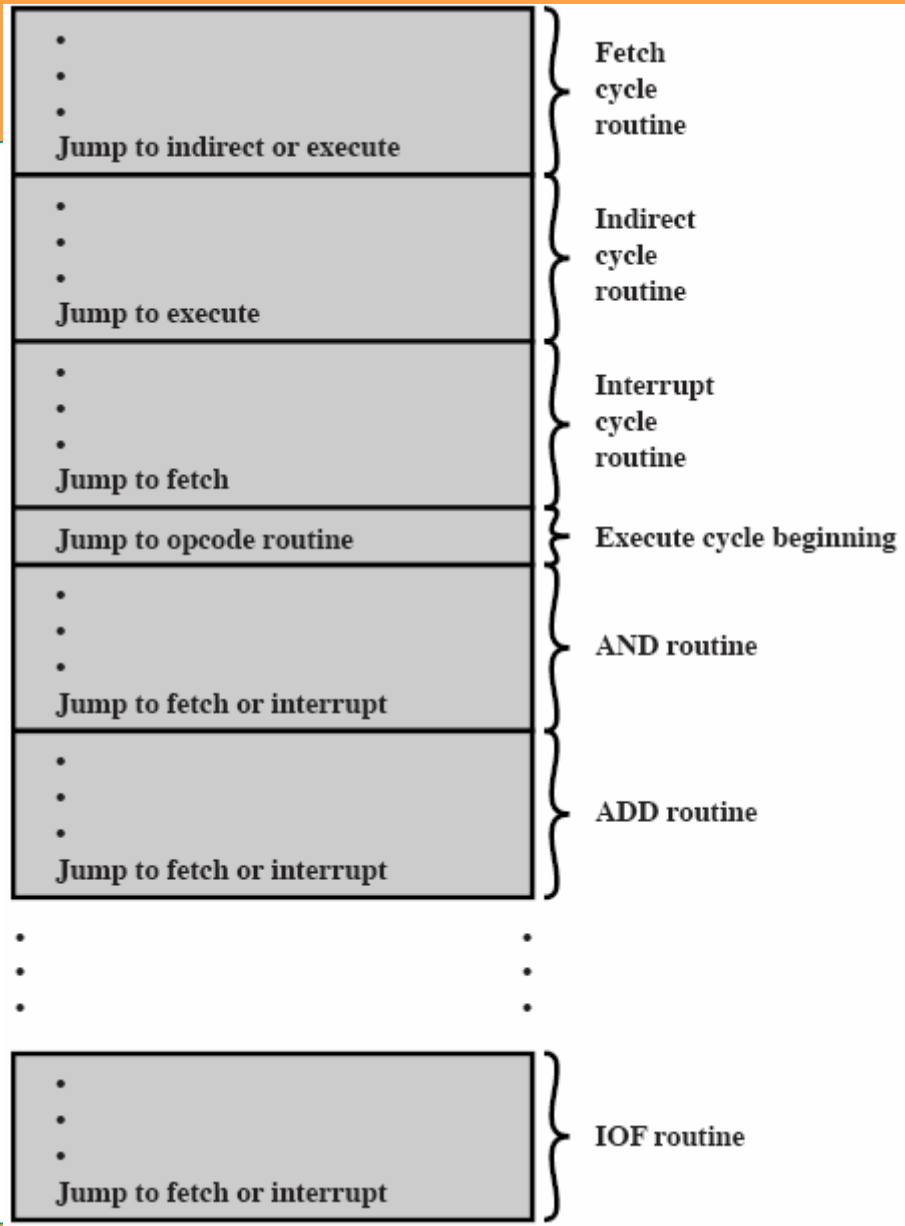
Mikro-ohjelmoitu ohjaus

- **Idea 1951: Wilkes Microprogrammed Control Execution Engine**
  - Suorituta konekäsky yksi mikrokäsky kerrallaan generoimalla suoritusaikana tarvittavat ohjaussignaalit
  - Tulkitse mikrokäskyt ohjelmallisesti ohjaussignaaleiksi
- **Mikro-operaatiot kuvattu kontrollimuistissa mikrokäskyinä**
  - Laiteohjelmisto (firmware)
- **Kukin mikrokäsky muodostuu kahdesta osasta**
  - Mitä suoritetaan tulevalla syklillä?
    - Mikrokäskystä käy ilmi tarvittavat ohjaussignaalit
    - Toimita ohjaussignaalit piireille
  - Mikä mikrokäsky tulkitaan tämän jälkeen?
    - Oletus: seuraavasta kontrollimuistin muistipaikasta

Sta06 Table 16.1

Mikrokoodi

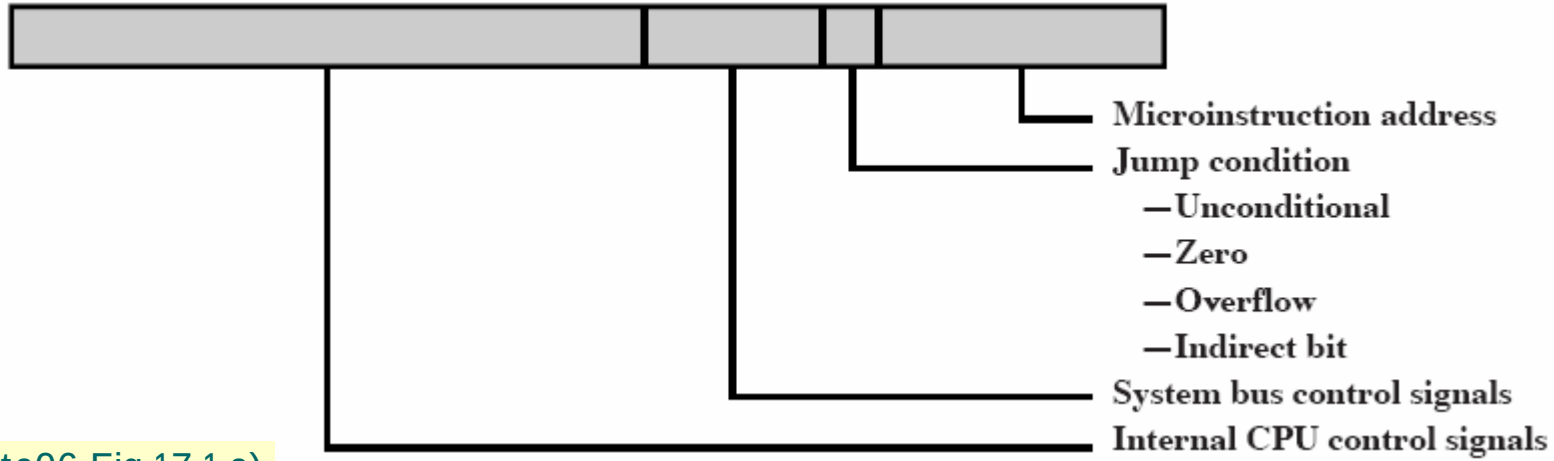
- Kutakin CPU:n käskysyklin vaihetta vastaa ”alirutiini”, joka suoritetaan käskysyklin aikana
- Esim. ROM -muistissa



(Sta06 Fig 17.2)

Horisontaalinen mikrokäsky

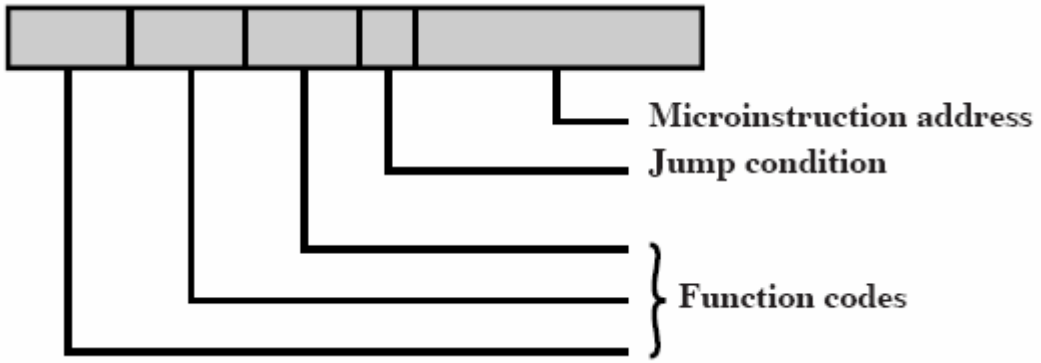
- Kaikki mahdolliset ohjaussignaalit kuvattu bittikarttana jokaisessa mikrokäskyssä
  - Yksi bitti per mahd. ohjaussignaali (1=generoi, 0=älä generoi)
  - Jos paljon erilaisia, mikrokäskystä tulee pitkä
- Kukin mikrokäsky myös ehdollinen hyppykäsky
  - Mitä statusbittejä tutkittava
  - Seuraavan mikrokäskyn osoite



(Sta06 Fig 17.1 a)

Vertikaalinen mikrokäsky

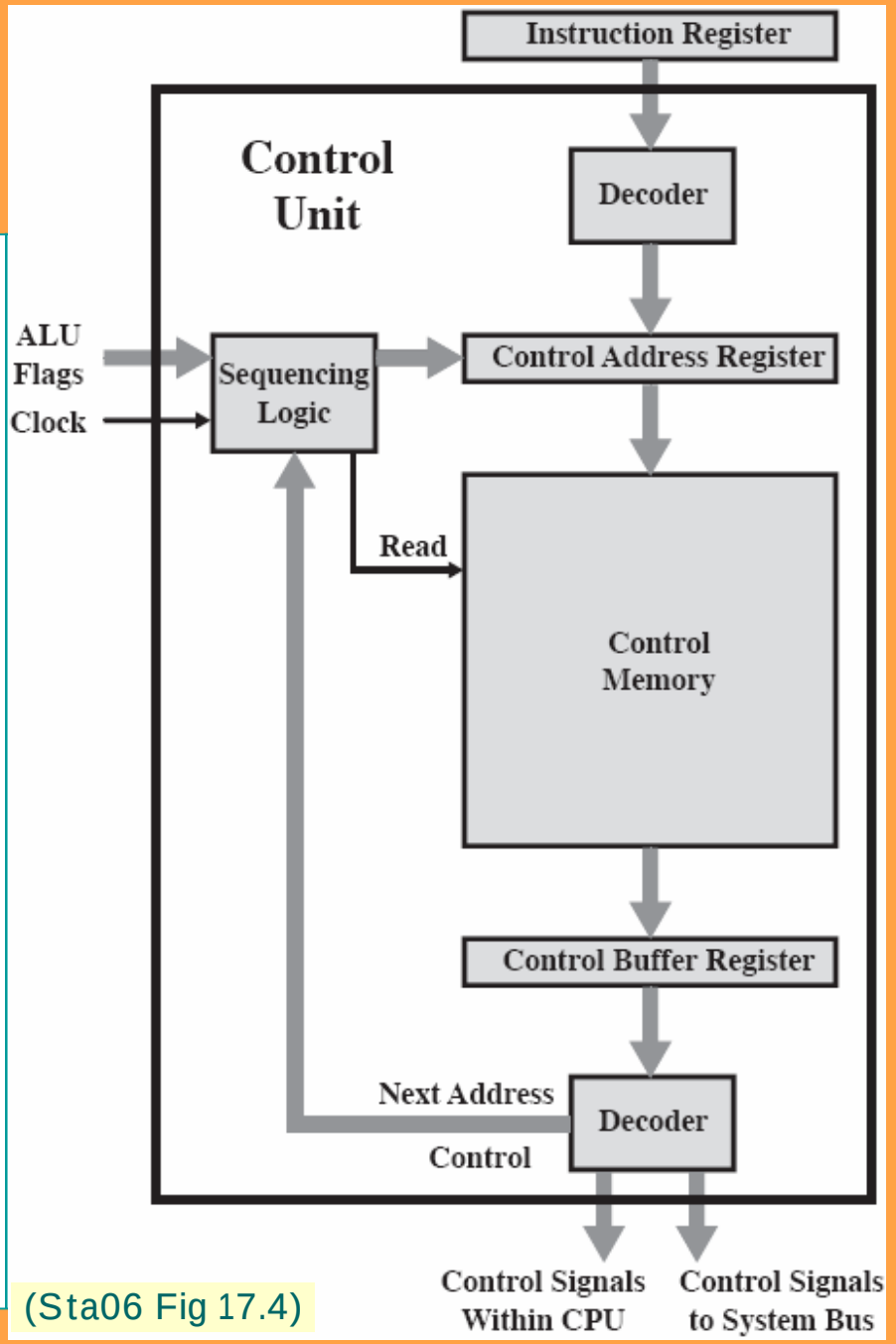
- Ohjaussignaalit koodattu toimintonumeroiksi
- Dekoodaa takaisin ohjausignaaleiksi suoritusaikana
- Lyhyemmät käskyt, mutta dekoodaus vie aikaa
- Kukin mikrokäsky myös ehdollinen hyppykäsky
  - Mitä statusbittejä tutkittava
  - Seuraavan mikrokäskyn osoite



(Sta06 Fig 17.1 b)

Ohjausyksikkö = Execution Engine

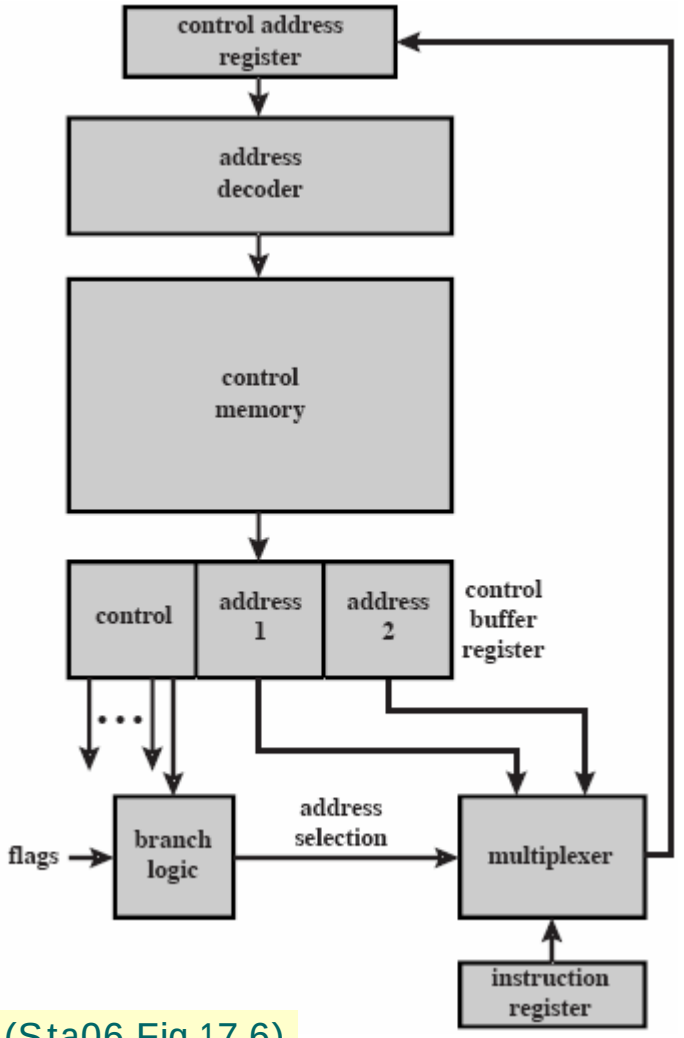
- **Control Address Register, CAR**
  - Mikä mikrokäsky seuraavaksi?
    ~ käskyosoitin, “MiPC”
- **Kontrollimuisti**
  - Mikrokäskyt
  - Nouto-suoritus-keskeytys
- **Control Buffer Register, CBR**
  - Rekisteri mikrokäskyn tulkintaa varten
    ~ käskyrekisteri, “MiIR”
  - Tulkitse ja lähetä ohjaussignaalit piireille
- **Sequencing Logic**
  - Seuraavaksi suoritettavan mikrokäskyn osoite CAR:iin



(Sta06 Fig 17.4)

Mikä mikrokäsky seuraavaksi?

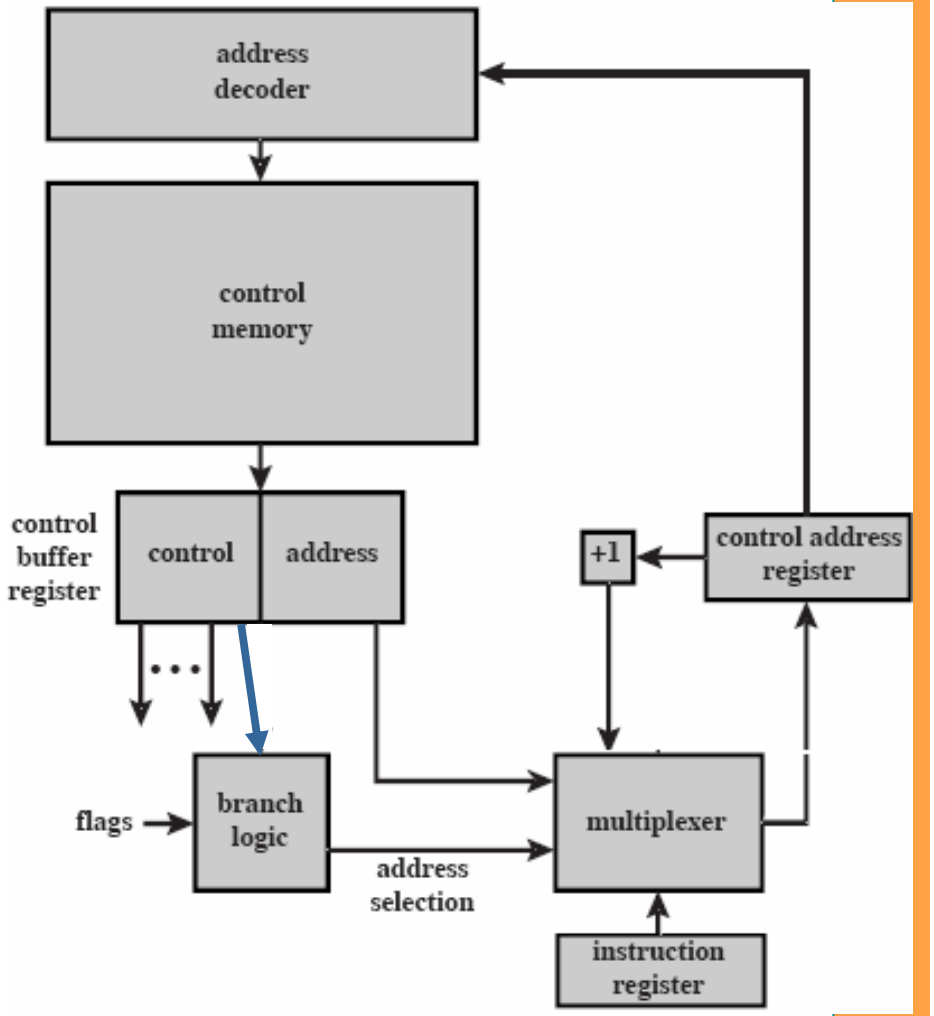
- a) Kerrottu eksplisiittisesti
- Käskyssä 2 osoitetta
  - Lisäksi kerrottu lipuke, jota tutkittava
  - Hae uusi jommasta kummasta paikasta
  - Usein heti seuraavasta kontrollimuistin muistipaikasta
    - Miksi tallettaa?
    - Ei aikaa laskemiseen!



(Sta06 Fig 17.6)

Mikä mikrokäsky seuraavaksi?

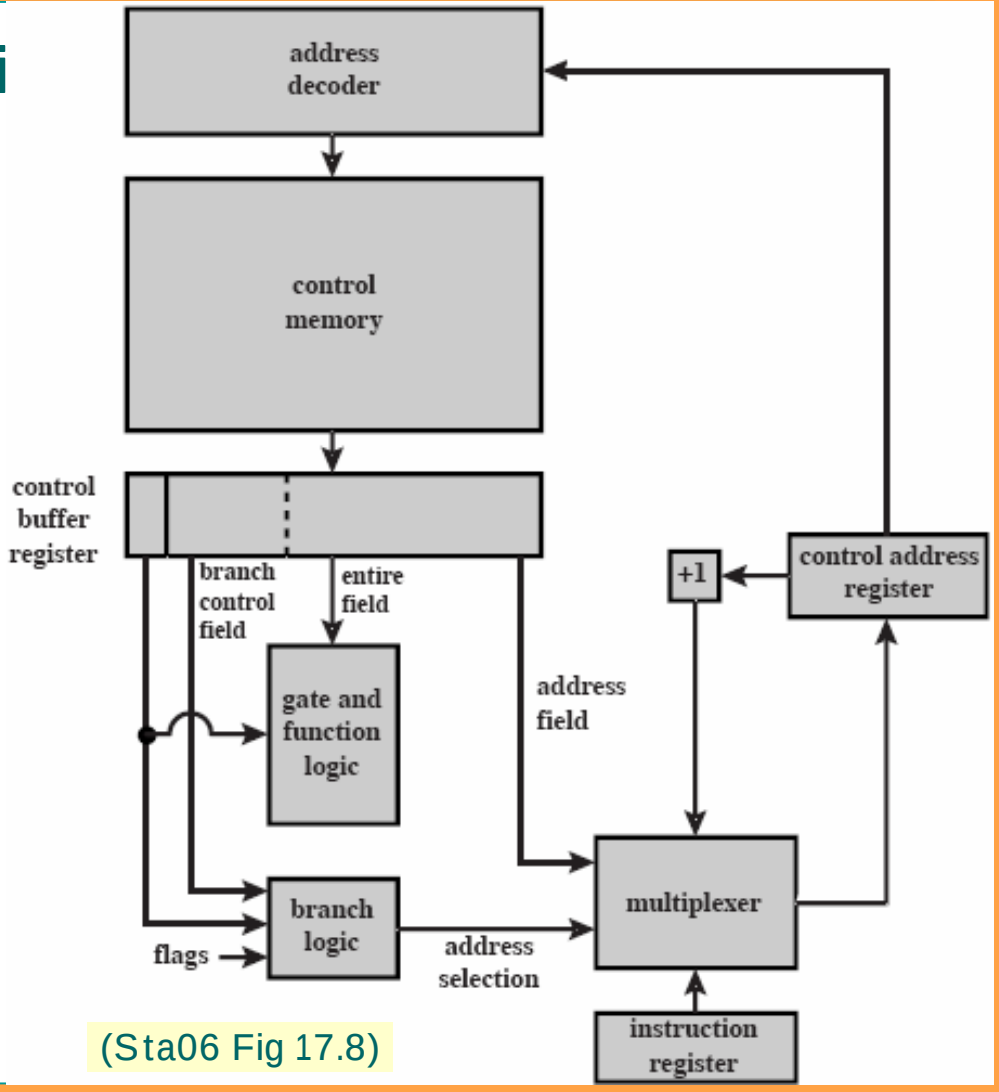
- b) Implisiittinen oletus: seuraavasta kontrollimuistin osoitteesta
- Käskyssä 1 osoite
  - Lisäksi kerrottu lipuke, jota tutkittava
  - Jos ehto=1, käytä käskyn osoiteosaa
- Osoiteosaa ei taaskaan käytetä aina
  - Useimmiten hukkatilaa



(Sta06 Fig 17.7)

Mikä mikrokäsky seuraavaksi?

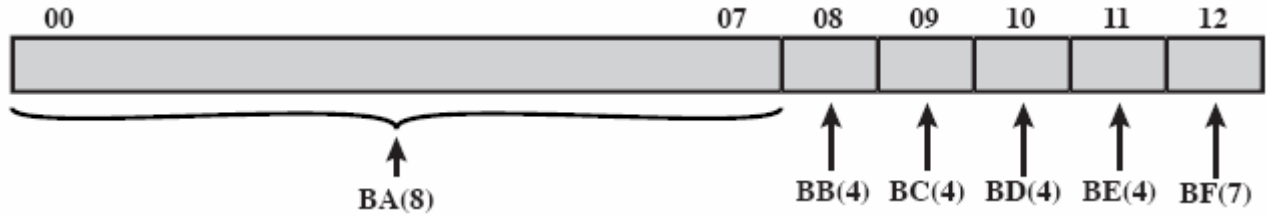
- **c) Vaihteleva formaatti**
  - Osa biteistä voidaan tulkita kahdella tavalla
  - 1 b: Käskyssä osoite/ei
  - Vain hyppykäskyssä mukana osoite
  - Hyppykäskyissä ei ole ohjaussignaaleja
  - Jos hyppy, pitääkin suorittaa kaksi käskyä yhden sijasta
    - Hukkaa aikaa?
    - Säästää tilaa?



(Sta06 Fig 17.8)

Mikä mikrokäsky seuraavaksi?

- **d) Osoitteen generointi suoritusaikana**
- Miten päästä oikean ALU-alirutiinin alkuun?
  - Konekäskyn suoritussignaalit riippuvat operaatiosta
- Generoi ensimmäisen mikrokäskyn osoite operaatio-koodista (mapping + combining/adding)
  - Eniten merkitsevät bitit op-koodista
  - Vähiten merkitsevät nollia tai generoitu nykytilan (status) perusteella
  - Esim: IBM 3033 CAR, 13 b:n osoite
    - Op-koodista 8 bittiä -> alirutiini max 32 käskyä
    - loput 5 bittiä statusbittejä tutkimalla



(Sta06 Fig 17.9)

Mikä mikrokäsky seuraavaksi?

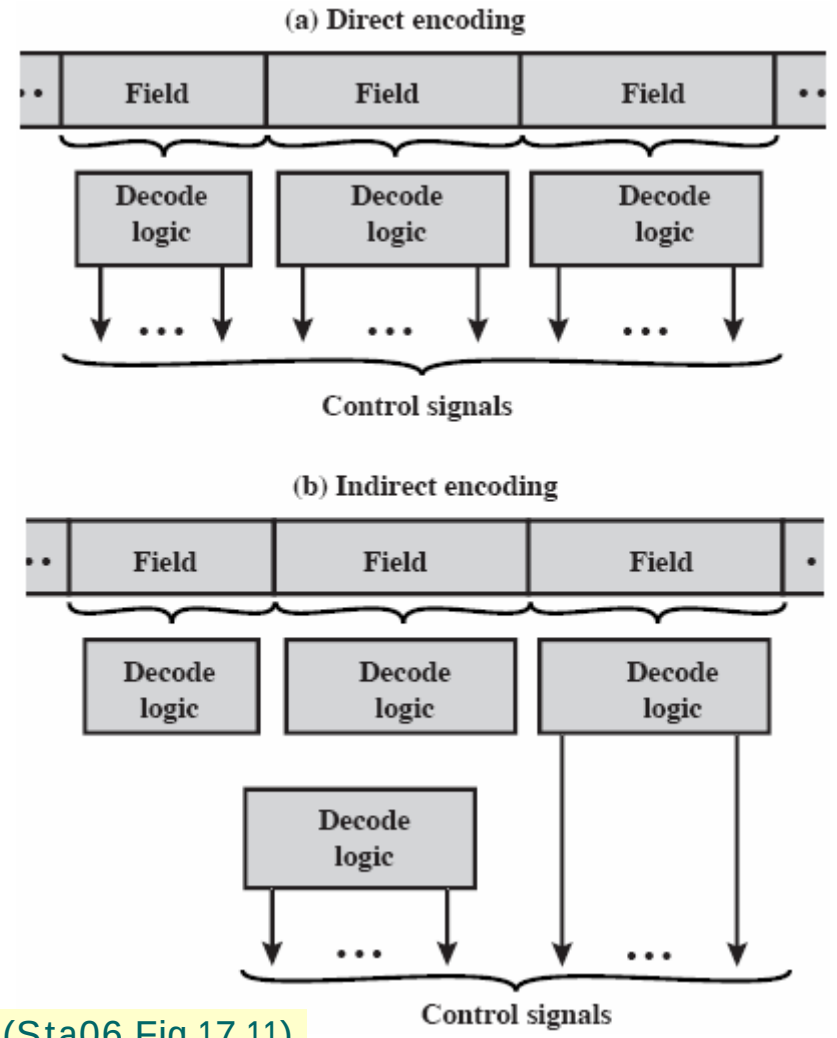
- **e) Aliohjelmakutsu ja paluu** (Residual control)
- Alirutiinin kutsu mikrotasolla
  - Ei ympäristön vaihtoa, vain yksi taso
  - Sisäkkäiset kutsut eivät sallittuja
  - Esim: LSI-11, 22b:n mikrokäsky
    - Kontrollimuisti 2048 käskyä, 11 b:n osoitteet
    - Operaatiokoodi määrää ens. mikrokäskyn osoitteen
    - Oletus seuraava CAR • CAR+1
    - Jokaisessa mikrokäskyssä bitti: alirutiinin kutsu/ei
      Alirutiinin kutsukäsky:
      - talleta paluuosoite paluurekisteriin (vain yksi)
      - hyppää alirutiiniin (osoite mikrokäskyssä)
    - Paluukäsky: ota osoite paluurekisteristä

Mikrokäskyn koodaus

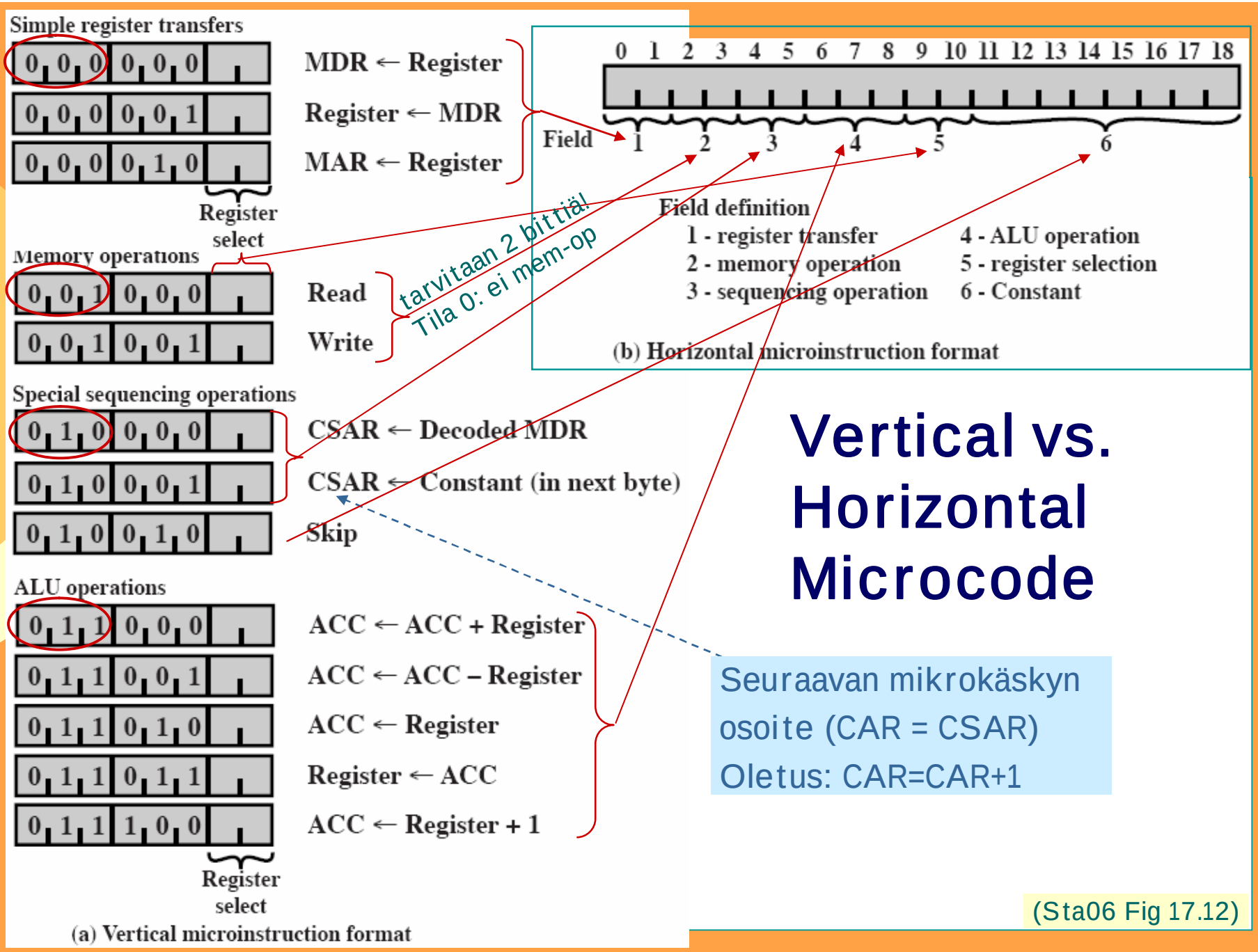
- **Horisontaalinen? Vertikaalinen?**
  - Horisontaalinen: nopea tulkinta
  - Vertikaalinen: vähemmän bittejä
- **Usein kompromissi eli sekamuoto**
  - Mikrokäsky jaettu kenttiin, kukin kenttä määrää tiettyjen toimintojen ohjaussignaalit
  - Keskenään toisensa poissulkevat toiminnot koodattavissa samaan kenttään
  - Koodaus purettava suoritusaikana ohjaussignaaleiksi
- **Kun useita koodauksia yhden sijasta, toteutus helpompaa**
  - Useita yksinkertaisia dekoodereita

Mikrokäskyn koodaus

- **Toimintojen mukaan** (functional encoding)
  - Kukin kenttä kontrolloi tiettyä toimintaa
    - Lataa akkurekisteristä
    - Lataa muistista
    - Lataa ...
- **Resurssien mukaan** (resource encoding)
  - Kukin kenttä kontrolloi tiettyä resurssia
    - Lataa akkurekisteristä
    - Talleta akkurekisteriin
    - Lisää akkurekisteriin



(Sta06 Fig 17.11)



Vertical vs. Horizontal Microcode

Seuraavan mikrokäskyn osoite (CAR = CSAR)
Oletus: CAR=CAR+1

(Sta06 Fig 17.12)

Miksi mikro-ohjelmoituna?

vaikka **hitaampi suoritus kuin langoitetulla logiikalla**

- **Suunnittelun helppous ja joustavuus**
  - Muutokset/laajennukset mukaan vaikka loppumetreillä
  - Vanhaan laitteistoon vain kontrollimuistin päivitys
    - Omalla lastullaan vähän vanhemmissa koneissa
  - Tätä varten omat kehitysympäristönsä
- **Taaksepäin yhteensopivuus**
  - Vanha käskykanta säilytettävissä vaivatta
  - Lisää vain uudet mikrokäskyt uusille käskyille
- **Yleisyys**
  - Yksi laitteisto, useita erilaisia käskykantoja
  - Yksi käskykanta, useita erilaisia organisointeja

Kertauskysymyksiä

- Langoitettu vs. mikro-ohjelmoitu toteutus?
- Kuinka mikrokäskyn osoite määräytyy?
- Mihin tarvitaan kontrollimuistia?
- Horisontaalinen vs. vertikaalinen mikrokäsky?
- Miksi ei mikro-ohjelmointia?
- IA-64 kontrolli vs. mikro-ohjelmointi vs. langoitettu kontrolli?

Tietokoneen rakenne

Pääotsikoita olivat

- Digitaalilogiikka
- Väylät, välimuisti, keskusmuisti
- Virtuaalimuistin osoitemuunnos, TLB
- ALU: kokonais- ja liukulukuaritmetiikka
- Käskykannoista: operaatiot ja osoittaminen
- CPU:n rakenne ja liukuhihna
- Hyppyjen ennustus, riippuvuudet
- RISC & superskalaari CPU, nimiriippuvuudet
- IA-64: Explicit Parallel Instruction Computing
- Langoitettu vs. mikro-ohjelmoitu ohjaus

-- The End --

STI Cell Power processor element
(a) major units
and
(b) pipeline

http://researchweb.watson.ibm.com/journal/rd/494/kahle2.gif