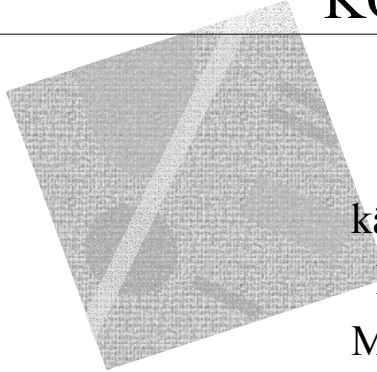


## Jakso 2

# TTK-91-tietokone ja sen KOKSI-simulaattori



Miksi TTK-91?

TTK-91:n rakenne ja  
käskykanta-arkkitehtuuri

Mikä on simulaattori?

Miten TTK-91-ohjelmia  
suoritetaan simulaattorissa?

## Miksi konekieltä?

- Koneen toiminnan ymmärtäminen
- Oman ohjelman toiminnan ymmärtäminen
- Koneenläheinen ohjelmointi
- Kääntäjän tekeminen
  - kääntäjä kääntää konekielelle lausekielisen ohjelman
- Ohjelman tehokkuus
  - osia ohjelmasta ohjelmoidaan suoraan konekielellä

# Miksi ei oikeaa konekieltä?

- Oikeat konekielet huomattavasti monimutkaisempia
  - niiden opetteluun tarvitaan oma kurssi
- Vaikeaa valita sopivinta
  - paljon erilaisia konekieliä
- Keskitytään vain opetuksen kannalta oleellisiin asioihin
  - tarvittaessa oikea konekieli 'helppo' oppia

10/05/2004

Copyright Teemu Kerola, S2003

3

# Tietokone TTK-91

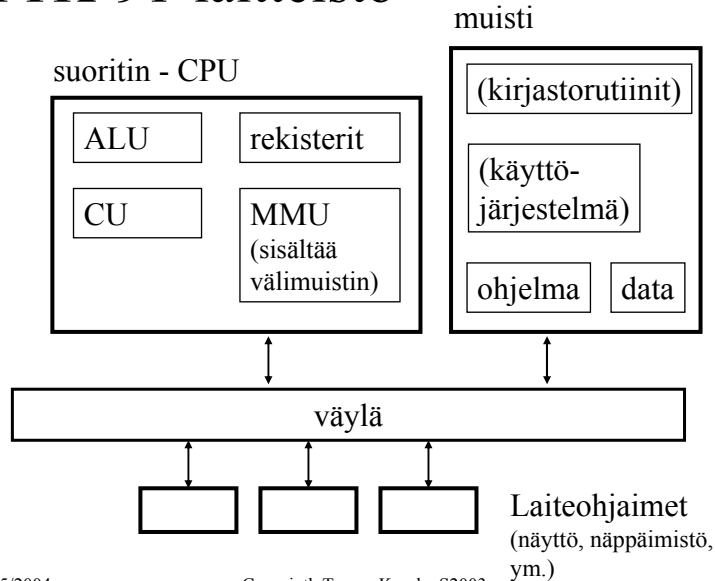
- Laitteisto, hardware (HW)
  - suoritin, muisti, väylät, oheislaitteiden liitännät
- Käskykanta - konekieliarkkitehtuuri
  - käyttöliittymä laitteistoon
  - konekäskyt, tiedon esitysmuodot, tietotyypit
- Symbolinen konekieli
  - luettavampi muoto konekielestä
  - kullakin symbolilla yksikäsitteiset arvot
- KOKSI-simulaattori
  - ohjelma, joka simuloi TTK-91-koneen laitteistoa
  - symbolisen konekielen kääntäjä
  - graafinen käyttöliittymä, debugger-ympäristö

10/05/2004

Copyright Teemu Kerola, S2003

4

# TTK-91-laitteisto



10/05/2004

Copyright Teemu Kerola, S2003

5

## TTK-91: rekisterit

- 8 yleisrekisteriä

ks. Kuva 4.1 [Häkk98]

- vain näitä rekistereitä voi koskettaa (suoraan) konekäskyillä
- kaikki laskenta tapahtuu rekistereiden avulla
  - vain 8 'muistipaikkaa' varsinaista laskentaa varten
- R0 työrekisteri
  - indeksirekisterinä == 0 (tietyissä tilanteissa R0:n käyttö tarkoittaa lukua 0 rekisterin R0 sisällön asemesta)
- R1-R5 työ- ja indeksirekistereitä
  - tyyppi riippuu rekisterin käytöstä konekäskyssä
- pino-osoitin SP (R6)
- ympäristöosoitin FP (R7)

Stack Pointer

Frame Pointer

10/05/2004

Copyright Teemu Kerola, S2003

6

# TTK-91: Kontrolliyksikkö (CU)

ks. Kuva 4.1 [Häkk98]

- PC - Program Counter, käskyosoitin
  - seuraavaksi suoritettavan konekäskyn osoite
- IR - Instruction Register, käskyrekisteri
  - suorituksessa oleva konekäsky
- TR - Temporary Register, apurekisteri
  - tilapäinen talletuspaikka käskyn suoritusajaksi
- SR - State Register, tilarekisteri
  - suorittimen tila ja rajoitukset tällä hetkellä

10/05/2004

Copyright Teemu Kerola, S2003

7

# TTK-91: Tilarekisteri SR <sup>(3)</sup>

- Tilatietoa siitä, mitä suorittimella tapahtui edellisen käskyn suorituksessa
  - virhetilanteet, poikkeukset ks. Kuva 4.1 [Häkk98]
  - konekäsky olikin käyttöjärjestelmän palvelupyynnö
  - vertailun tulos
- Tilatietoa siitä, mitä systemissä tapahtui viime aikoina
  - käsittelemättömät laitteiden antamat signaalit (laitekeskeytykset, device interrupts)
- Tilatietoa siitä, mitä suoritin saa tehdä jatkossa
  - etuoikeutettu tila?
  - (kaikki muistialueet ja kaikki käskyt sallittuja)
  - poikkeukset ja keskeytykset sallittuja vai ei?

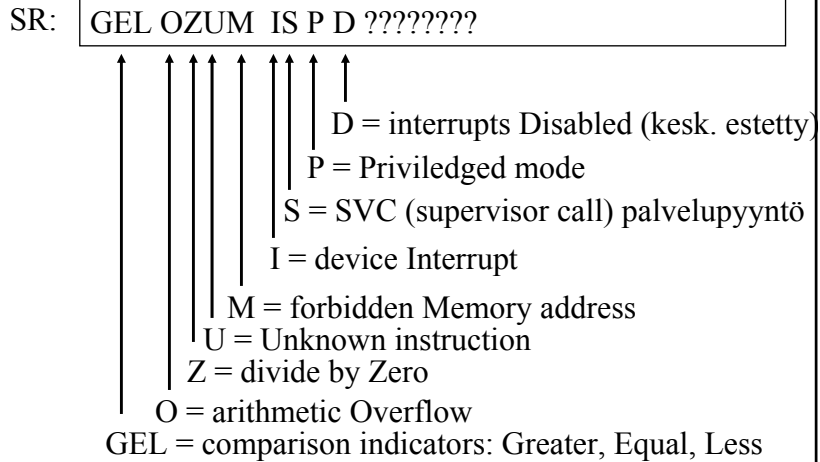
10/05/2004

Copyright Teemu Kerola, S2003

8

# Tilarekisteri SR (9)

32 bittiä (kunkin arvo 0 tai 1)



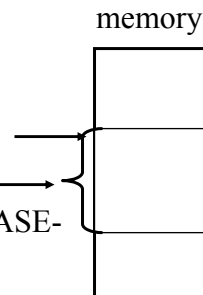
10/05/2004

Copyright Teemu Kerola, S2003

9

# TTK-91: Muistinhallintayksikkö (MMU)

- Muistiinviittausrekisterit ks. Kuva 4.1 [Häkk98]
  - MAR - Memory Address Register, muistiosoite
  - MBR - Memory Buffer Register, luettava/kirjoitettava arvo
- Käytössä oleva muistialue
  - vain tähän alueeseen voi nyt viitata
  - BASE-muistisegmentin alkuosoite
  - LIMIT-muistisegmentin koko
  - kaikki muistiosoitteet suhteellisia BASE-rekisterin arvoon
    - Esim. jos BASE = 8000, niin ohjelman osoite 34 viittaa muistiosoitteeseen 8034
  - käyttöjärjestelmä asettaa ja valvoo



10/05/2004

Copyright Teemu Kerola, S2003

10

# TTK-91: Käskykanta

- Tietotyypit
- Konekäskyjen tyypit
- Konekäskyn rakenne
  - montako bittiä, minkälainen sisäinen rakenne
- Muistissa olevan tiedon osoitustavat
  - konekielessä
  - symbolisessa konekielessä
- Operaatiot

10/05/2004

Copyright Teemu Kerola, S2003

11

# TTK-91: tietotyypit <sup>(2)</sup>

- 32 bittinen kokonaisluku
  - noin 10 desimaalilukumeroinen luku
- EI:
  - liukulukuja
  - merkkejä
  - totuusarvoja
  - ...

10/05/2004

Copyright Teemu Kerola, S2003

12

# TTK-91: käskytyypit

- Aina 2 operandia itse käskyssä
  - aina ei molemmilla ole merkitystä
    - JUMP vain yksi operandi, hypyn osoite
    - NOP ei operandeja lainkaan
- Käsky aina 32 bittiä
- Ensimmäinen operandi aina rekisterissä
  - Käytössä työ- ja indeksirekisterit R0-R5 sekä erikoisrekisterit SP, FT
- Toinen operandi muistissa tai rekisterissä
  - luku rekisteristä on nopeampaa kuin muistista hakeminen
- ALU-operaatioiden tulos rekisteriin
  - korvaa 1. operandin arvon !

10/05/2004

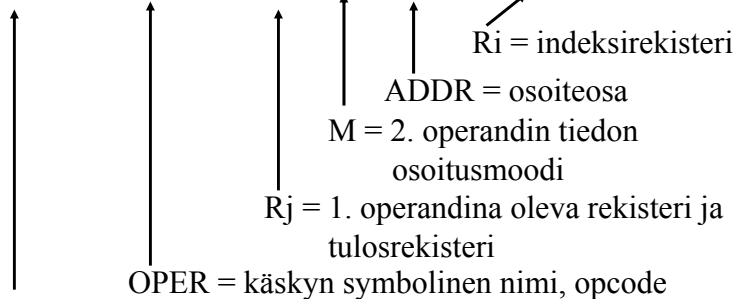
Copyright Teemu Kerola, S2003

13

## Symbolinen konekieli <sup>(6)</sup>

Loop: ADD R4, @TAULU(R1)

viite: OPER R<sub>j</sub>, M ADDR(R<sub>i</sub>)



viite = käskyn (symbolinen) osoite

- Suora vastaavuus konekieleen
  - yksinkertainen assembler-käännös

10/05/2004

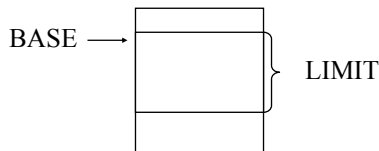
Copyright Teemu Kerola, S2003

14

# Symbolinen konekieli

- Symbolien vastaavuus 1:1 kaikkialla
  - viite = käskyn muistiosoite (eli tietyn tyyppinen vakio)
  - operaatiokoodi, opcode = vakio
    - kullekin operaatiolle oma koodinsa
  - osoitekentän symboli = vakio tai muistiosoite
    - kenttään voi kirjoittaa joko symbolin tai arvon!

Kaikki ohjelmassa käytetyt muistiosoitteet ovat suhteellisia ohjelmalle varatun muistitilan alkuun (= BASE) nähden eli arvoalueella  $[0, \text{LIMIT}-1]$



10/05/2004

Copyright Teemu Kerola, S2003

15

# Symbolinen konekieli (2)

- Osoitusmoodi: monimutkaisempi vastaavuus
  - konekielessä 3 moodia
    - **vakio ja/tai rekisteri** (tieto konekäskyssä + rekisterissä eli ei tarvita muistista haku)
    - **indeksoitu ja epäsuora indeksoitu** (tieto muistissa; haettava muistista 1 tai 2 kertaa)
  - symbolisessa konekielessä 8 moodia
    - helpottavat ohjelmointia
    - toteutettu konekielen 3 moodin avulla

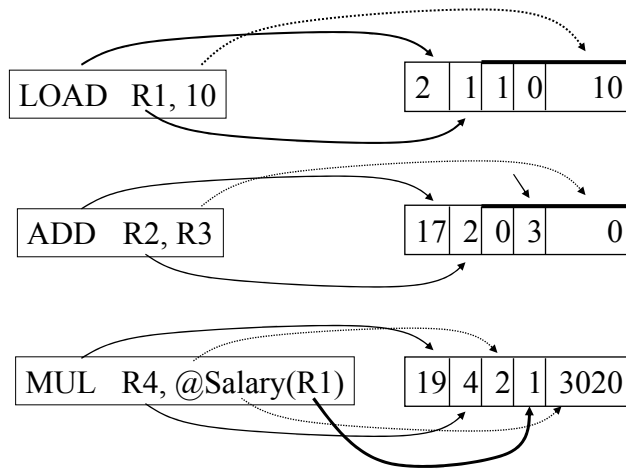
10/05/2004

Copyright Teemu Kerola, S2003

16



# Symbolinen konekieli vs. konekieli <sup>(3)</sup>



10/05/2004

Copyright Teemu Kerola, S2003

17

## Tiedon osoitusmuodot symbolisessa konekielessä

- 8 eri osoitusmoodia (vain 2. operandille!)
- Tekstuaalisesti koodattuna

– osoitusmoodi

LOAD R1, @Field1(R3)

- = vakio [+ rekisterin arvo]
- tyhjä arvo rekisterissä tai muistissa
- @ epäsuora viite muistiin = muistissa tai rekisterissä on vasta arvon osoite

LOAD R2, =100

R2 <= 100

LOAD R2, 100

R2 <= MEM[100]

LOAD R2, @100

R2 <= MEM[MEM[100]]

10/05/2004

Copyright Teemu Kerola, S2003

18

## Indeksointi <sup>(2)</sup>

```
LOAD R4,=Tbl(R3)
LOAD R4, Tbl(R3)
LOAD R4, @Tbl(R3)
```

- Laske aina ensin tehollinen muistiosoite (effective address, EA):  $EA = Tbl + (R3) = 201$
- Sitten katso moodia ja tee niin monta muistinoutoa kun tarvitaan
  - ”=”: 0 kpl  $R4 \leftarrow 201$
  - ”tyhjä”: 1 kpl  $R4 \leftarrow Mem[201] = 11$
  - ”@”: 2 kpl  $R4 \leftarrow Mem[ Mem[201] ]$   
 $= Mem[ 11 ] = 300$

STORE käsky  $\Rightarrow$  1 kpl vähemmän noutoja ja yksi tallennus

## Lyhennysmerkintöjä:

- 0-arvoa ei yleensä kirjoiteta näkyviin
  - indeksirekisterinä R0 tai vakiona 0
- 2. rekisteri on itseasiassa aina indeksirekisteri:
  - LOAD R1,R2 (LOAD R1, =0(R2))
  - LOAD R1,@R2 (LOAD R1, 0(R2))
- indeksirekisterinä R0  $\Rightarrow$  indeksirekisterin arvo = 0  $\Rightarrow$  ei indeksointia
  - LOAD R1,10 (LOAD R1, 10(R0))



- **Rekistereille** on vain kaksi osoitustapaa:

- **LOAD R1, R2**       $R2 = 10$   
     $\Rightarrow$   $R1 \leq 10$   
– **LOAD R1, =0(R2)** täydellisesti kirjoitettuna

- **LOAD R1, @R2**       $R2 = 10, M[10] = 15,$   
     $\Rightarrow$   $R1 \leq 15$   
– **LOAD R1, 0(R2)** täydellisesti kirjoitettuna

- **Huom!** Vain kaksi rekisteriä käytössä.

~~LOAD R1, R2(R3)~~ on virheellinen käsky

## Muistinoutojen määrä

- **0 kpl**

- osoiteosassa
  - vakio: **LOAD R1, =10**, **LOAD R1, =sata**
  - 2. operandi rekisteri: **LOAD R2, R1**

- **1 kpl**

- osoiteosassa
  - muistipaikka: **LOAD R1, 10**, **LOAD R1, sata(R2)**
  - osoite rekisterissä: **LOAD R1, @R2**

- **2 kpl**

- osoiteosassa
  - osoite muistipaikkaan: **LOAD R1, @100(R3)**

# TTK-91 muistin osoitusmoodit

ks. lista sivulla 50  
[Häkk98]

rekisterit	
R0:	104
R1:	10
R2:	201
R3:	1
...	
SP=R6:	
FP=R7:	125

muisti- segmentti	
0:	
10:	200
11:	300
200:	6000
201:	11

LOAD R1, 10	; R1 ← 200
LOAD R1, =10	; R1 ← 10
LOAD R1, @10	; R1 ← 6000
LOAD R4, R2	; R4 ← 201
LOAD R4, @R2	; R4 ← 11
LOAD R5, =Tbl(R3)	; R5 ← 201
LOAD R5, Tbl(R3)	; R5 ← 11
LOAD R5, @Tbl(R3)	; R5 ← 300

LIMIT:

symboli- taulu	
Tbl:	200
X:	10
One:	1

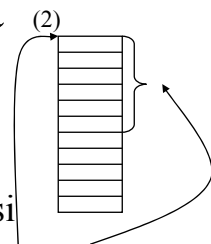
10/05/2004

Copyright Teemu Kerola, S2003

25

## Indeksoinnin käyttö taulukkojen ja tietueiden yhteydessä

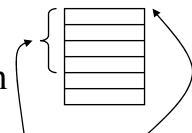
- Taulukot
  - taulukon alkuosoite vakiona
  - indeksirekisterissä taulukon indeksi



LOAD R5, Tbl(R3)

1854 14

- Tietueet
  - indeksirekisterissä tietueen alku
  - vakiona tietueen kentän suhteellinen osoite tietueen sisällä



LOAD R2, Salary(R5)

6 1244

10/05/2004

Copyright Teemu Kerola, S2003

26

# TTK-91 operaatiot

- Muistiinviittaukset
  - tavalliset: load & store
  - pino-operaatiot
- I/O käskyt
- Kokonaislukuoperaatiot
- Loogiset operaatiot totuusarvoille
- Bittien siirtokäskyt (shift instructions)
- Kontrollin siirtokäskyt
  - mistä löytyy seuraavaksi suoritettava käsky?
- Muut käskyt

10/05/2004

Copyright Teemu Kerola, S2003

27

# TTK-91-muistiinviittausoperaatiot

- LOAD LOAD R1, X LOAD R5, @ptrX
  - myös rekistereiden kopiointiin (Move-operaatio) LOAD R0, R2
- STORE STORE R2, X
  - aina muistiin tallennus STORE R3, Tbl(R4)
- PUSH, POP, PUSHR, POPR
  - aliohjelmien toteuttamista varten POP SP, R1 ; load ...
  - käsitellään myöhemmin PUSH SP, R1 ; store ...

10/05/2004

Copyright Teemu Kerola, S2003

28

# TTK-91: I/O- operaatiot

- IN

IN R3, =KBD

- lue arvo (kokonaisluku) rekisteriin annetulta laitteelta

- OUT

OUT R2, =CRT

- tulosta arvo (kokon. luku) rekisteristä annetulle laitteelle

- Laitteet?

- KBD - näppäimistö, stdin
- CRT - näyttö, stdout
- ei muita! (ei levyä, ei verkkoa, ...)

10/05/2004

Copyright Teemu Kerola, S2003

29

# TTK-91: kokonaislukuoperaatiot

- LOAD ('move')

LOAD R3, R1 ; R3 ← R1

- ADD, SUB

ADD R3, R1 ; R3 ← R3+R1

SUB R3, =1 ; R3 ← R3 - 1

- MUL

MUL R3, Tbl(R1) ; R3 ← R3 \* Mem[Tbl+R1]

- DIV, MOD

```
LOAD R1,=14
DIV   R1,=3      ; R1 ← 4
LOAD R1,=14
MOD  R1,=3      ; R1 ← 2
```

10/05/2004

Copyright Teemu Kerola, S2003

30

# TTK-91

## loogiset operaatiot <sup>(4)</sup>

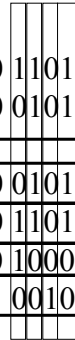
- NOT, AND, OR, XOR

- kaikille 32 bitille
- yksi bitti kerrallaan



LOAD R1, =13 ; R1 = 000...000 1101  
LOAD R2, =5 ; R2 = 000...000 0101

AND R1,R2	; R1 = 000...000	0101
OR R1,R2	; R1 = 000...000	1101
XOR R1,R2	; R1 = 000...000	1000
NOT R1	; R1 = 111...111	0010



# TTK-91:

## bittien siirtokäskyt

- SHL, SHR

- siirrä bittejä vasemmalle tai oikealle
- täytä nolilla

LOAD R1,=5	;R1 = 000...000	00101 = 5
SHL R1,=1	;R1 = 000...000	01010 = 10



- positiivisilla luvuilla yhden bitin siirto vasemmalle on sama kuin 2:lla kertominen!
- positiivisilla luvuilla yhden bitin siirto oikealle on sama kuin 2:lla jakaminen!

LOAD R1,=5	;R1 = 000...000	00101 = 5
SHR R1,=1	;R1 = 000...000	00010 = 2



# TTK-91:

## kontrollin siirtokäskyt

- JUMP `JUMP Loop`
- COMP `COMP R3, =27` `COMP R3, X`
  - asettaa **tilarekisteriin SR** vertailun tuloksen: L, E tai G
- JLES, JEQU, JGRE, JNLE, JNEQU, JNGRE
  - perustuu tilarekisterin tietoon eli `JGRE Loop`  
viimeksi suoritettuun COMP-käskyyn
- JNEG, JZER, JPOS, JNNEG, JNZER, JNPOS
  - perustuu annetun rekisterin arvoon `JPOS R1, Loop`
- CALL, EXIT `(käsitellään myöhemmin)`
- SVC `SVC SP, =HALT ;ohjelman suoritus päättyy`

10/05/2004

Copyright Teemu Kerola, S2003

33

## TTK-91 muut käskyt

- NOP `NOP`
  - No Operation, tyhjä käsky, älä tee mitään
  - varaa kuitenkin muistia yhden sanan (32 bittiä)
  - suoritetaan samoin kuin muutkin käskyt

10/05/2004

Copyright Teemu Kerola, S2003

34

## TTK-91 assembler- kääntäjän ohjaukaskäskyt (4)

- Eivät generoi lainkaan konekäskyjä `Sata EQU 100`
- EQU - Equal `LOAD R1,=Sata`
  - antaa arvon symbolille **symbolitauluun**
- DC - data constant `X DC 50`
  - varaa yhden sanan tilaa muistista ja antaa sille arvon, antaa arvon symbolille (symbolitauluun!) `LOAD R1,X`
  - esim. muuttujan tai ison vakion määrittely
- DS - data segment `Tbl DS 200`
  - varaa monta sanaa tilaa muistista, antaa arvon symbolille
  - alkuarvot ovat epämääräisiä! `LOAD R3,Tbl(R1)`
  - Esim. taulukon tai tietueen tilan varaus

10/05/2004

Copyright Teemu Kerola, S2003

35

## TTK-91 symbolinen konekieliohjelma

```
hello.k91 X DC 13
          Y DC 15

MAIN LOAD R1, X
      ADD R1, Y
      OUT R1, =CRT
      SVC SP, =HALT
```

10/05/2004

Copyright Teemu Kerola, S2003

36

# TTK-91 symbolisia konekäskyesimerkkejä <sup>(10)</sup>

- Miten toimivat seuraavat käskyt?

```

LOAD R2, @100      ;R2 ← 200
ADD R2, 101 (R3)   ;R2 ← R2 +100 = 105
DIV R1, R3         ;R1 ← 0
LOAD R2, =100(R0)  ;R2 ← 100
LOAD R0, @101(R3) ;R0 ← 101
    
```

	regs		mem
R0:	2	100:	101
R1:	1	101:	200
R2:	5	102:	101
R3:	2	103:	100

10/05/2004

Copyright Teemu Kerola, S2003

37

# TTK-91 symbolisia konekäskyesimerkkejä <sup>(10)</sup>

- Entä miten toimivat seuraavat käskyt?

```

LOAD R2, @Xptrptr ;R2 ← 200
ADD R2, Xptr (R3) ;R2 ← R2 +100 = 105
DIV R2, R3        ;R2 ← 2
LOAD R2, =Tbl(R1) ;R2 ← 101
LOAD R2, Sum(R4) ;R2 ← 101
    
```

	regs		mem		symbol
R1:	1	100:	101	Tbl =	100
R2:	5	101:	200	X =	200
R3:	2	102:	101	Xptr =	101
R4:	100	103:	100	Xptrptr =	100
				Sum =	2

10/05/2004

Copyright Teemu Kerola, S2003

38

# TTK-91 symbolinen konekieli ohjelma

sum.k91

; sum - laske annettuja lukuja yhteen, luku 0 on loppumerkki

Luku DC 0 ; nykyinen luku, alkuarvo 0

Summa DC 0 ; nykyinen summa, alkuarvo 0

Sum IN R1, =KBD ; ohjelma Sum alkaa käskystä 0

STORE R1, Luku

JZER R1, Done ; luvut loppu?

LOAD R1, Summa ; Summa <- Summa+Luku

ADD R1, Luku

STORE R1, Summa ; summa muuttujassa, ei rekisterissa?

JUMP Sum

Done LOAD R1, Summa ; tulosta summa ja lopeta

OUT R1, =CRT

SVC SP, =HALT

10/05/2004

Copyright Teemu Kerola, S2003

39

## Konekielinen esitys

31

24 23 -21 20 19 18- 1615

0

OPER	Rj	M	Ri	ADDR
------	----	---	----	------

- Kullekin käskylle varattu 32 bittiä
  - OPER = operaatiokoodi (8 bittiä)
  - Rj = 1. operandina oleva rekisteri (3 bittiä)
  - Ri = käytetty indeksirekisteri (3 bittiä)
    - R0 = ei indeksointia
  - M = 2. operandin osoitusmoodi: 00, 01, 10
    - vakio, arvo muistipaikasta, osoite muistipaikassa
  - osoiteosa ADDR (16 bittiä)

10/05/2004

Copyright Teemu Kerola, S2003

40

# KOKSI

## TTK-91 -koneen simulaattori (7)

- Toimii kuten oikea kone toimisi
- Graafinen käyttöliittymä
- I/O vain käyttöliittymän kautta
- Ohjelmien valinta ("lataus"), käynnös ja suoritus
- Ohjelmien editointi ks. sum.k91
  - myös mikä tahansa tekstieditori kelpaa!
- Käsky kerrallaan suoritus mahdollinen
- Käsky kerrallaan, kommentoinnin kera

10/05/2004

Copyright Teemu Kerola, S2003

41

# KOKSI

## TTK-91 -koneen simulaattori

- Käytettävissä (DOS, W95, W98, W-NT, W2000)
  - laitoksen koneissa
  - kotona <http://www.cs.Helsinki.FI/u/kerola/tito/>
- Installoi itse kotihakemistoosi (n. 120 KB)
  - kopioi zip-tiedosto ja pura se koksi-hakemistoon
  - editoi koksi.cfg tiedostoon editorin polku

Esim: c:\windows\command\edit.com
- Ohjelmatiedostojen (hello.k91 jne) tulee olla samassa hakemistossa kuin simulaattorin (koksi.exe)
  - käynnistä (esim.) klikkaamalla koksi.exe

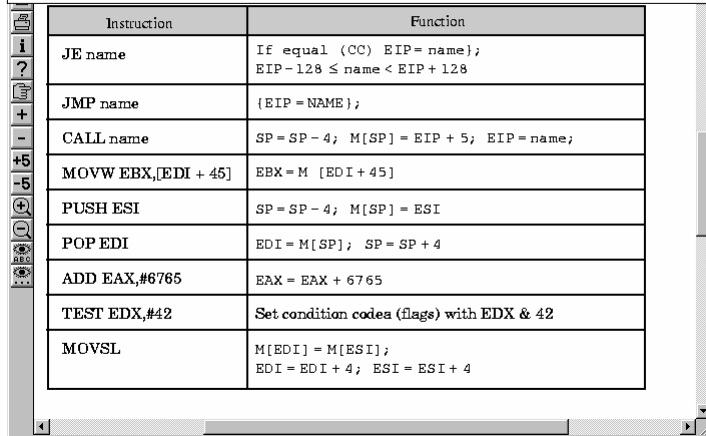
10/05/2004

Copyright Teemu Kerola, S2003

42

# -- Jakson 2 loppu --

Some typical 80x86 intructions and their function



Instruction	Function
JE name	If equal (CC) EIP = name); EIP - 128 ≤ name < EIP + 128
JMP name	{EIP = NAME};
CALL name	SP = SP - 4; M[SP] = EIP + 5; EIP = name;
MOVW EBX,[EDI + 45]	EBX = M [EDI + 45]
PUSH ESI	SP = SP - 4; M[SP] = ESI
POP EDI	EDI = M[SP]; SP = SP + 4
ADD EAX,#6765	EAX = EAX + 6765
TEST EDX,#42	Set condition codea (flags) with EDX & 42
MOVSL	M[EDI] = M[ESI]; EDI = EDI + 4; ESI = ESI + 4

Fig. 3.32 [PaHe98]