Luento 2
TTK-91 tietokone ja sen KOKSI simulaattori

Miksi TTK-91?

TTK-91 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

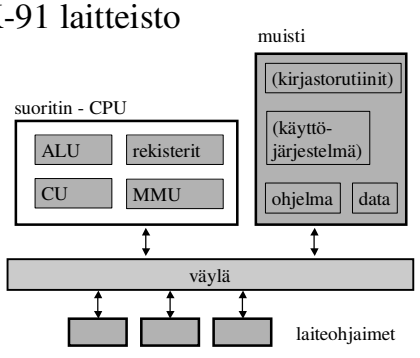
Miksi ei oikeaa konekieltä?

- 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

Tietokone TTK-91 (4)

- 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

TTK-91 laitteisto



TTK-91 rekisterit

- 8 yleisrekisteriä — ks. Kuva 4.1 [Häkk98]
  - vain näitä rekistereitä voi koskettaa (suoraan) konekäskyillä
  - kaikki laskenta tapahtuu rekistereiden avulla
  - R0 työrekisteri
    - indeksirekisterinä == 0 (tietyssä tilanteessä R0:n käyttö tarkoittaa lukua 0 rekisterin R0 sisällön asemesta)
  - R1-R5 työ- ja indeksirekistereitä
    - tyyppi riippuu konekäskystä
  - pino-osoitin SP (R6) — Stack Pointer
  - ympäristöosoitin FP (R7) — Frame Pointer

TTK-91 Kontrolliyksikkö (CU)

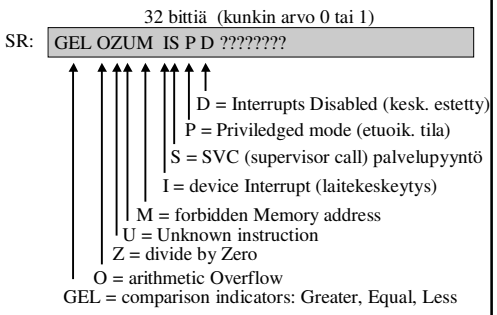
ks. Kuva 4.1 [Häkk98]

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

TTK-91 Tilarekisteri SR (3)

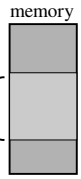
- Tilatietoa siitä, mitä suorittimella tapahtui edellisen käskyn suorituksessa
  - virhetilanteet, poikkeukset ks. Kuva 4.1 [Häkk98]
  - konekäsky olikin käyttöjärjestelmän palvelupyyntö
  - vertailun tulos
- Tilatietoa siitä, mitä systeemissä tapahtui viime aikoina
  - käsittelemättömät laitteiden antamat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

Tilarekisteri SR (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
  - käyttöjärjestelmä asettaa ja valvoo



TTK-91 Käskykanta

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

TTK-91 tietotyypit (2)

- 32 bittinen kokonaisluku
  - noin 10 desimaalinumeroinen luku
- EI:
  - liukulukuja
  - merkkejä
  - totuusarvoja
  - …

TTK-91 käskytyypit

- Aina 2 operandia itse käskyssä
  - aina ei molemmilla ole merkitystä
    - JUMP — vain yksi operandi, Ri+ADDR
    - NOP — ei operandeja lainkaan
- Käsky aina 32 bittiä
- Ensimmäinen operandi aina rekisterissä
- Toinen operandi muistissa tai rekisterissä
  - luku rekisteristä on nopeampaa kuin muistista hakeminen
- ALU-operaatioiden tulos rekisteriin
  - korvaa 1. operandin arvon!

Symbolinen konekieli (7)

```
LOOP:  ADD   R4, @TAULU(R1)
viite: OPER  Rj, M ADDR(Ri)
```

Ri = indeksirekisteri
ADDR = osoiteosa
M = 2. operandin osoitusmoodi
Rj = 1. operandina oleva rekisteri
OPER = käskyn symbolinen nimi, opcode
viite = käskyn (symbolinen) osoite

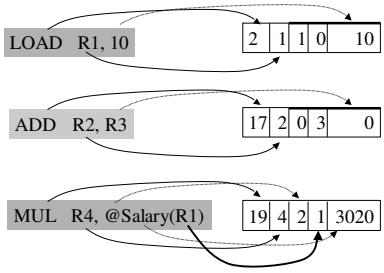
- Suora vastaavuus konekieleen
  - yksinkertainen assembler-käännös

Symbolinen konekieli

- Symbolien vastaavuus 1:1 kaikkialla
  - viite: muistiosoite
  - operaatiokoodi, opcode: vakio
  - osoitekentän symboli: vakio tai muistiosoite
    - kenttään voi kirjoittaa joko symbolin tai arvon!

  Kaikki muistiosoitteet suhteellisia BASE-osoitteeseen, eli arvoalueella [0, LIMIT-1]
- Osoitusmoodi: monimutkaisempi vastaavuus
  - konekielessä 2 moodia
    - indeksoitu, epäsuora indeksoitu
  - symbolisessa konekielessä 8 moodia
    - helpottavat ohjelmointia

Symbolinen konekieli vs. konekieli (3)

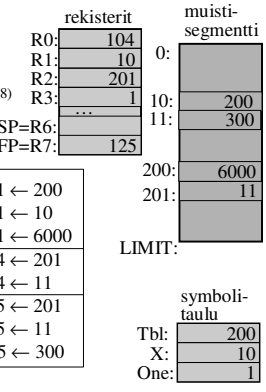


Operandin osoitusmuodot symbolisessa konekielessä

- 8 eri osoitusmoodia (vain 2. operandille!)
- Tekstuaalisesti koodattuna
  - osoitusmoodi — LOAD R1, @Field1(R3)
    - = vakio [+ rekisterin arvo]
    - tyhjä arvo rekisterissä tai muistissa
    - @ epäsuora viite muistiin
  - sulkumerkit rekisterin ympärillä
    - ei sulkuja käytä rekisterin arvoa sellaisenaan
    - sulut käytä rekisterin osoittamaa muistipaikan arvoa
  - 0-arvoa ei kirjoiteta näkyviin
    - indeksirekisterinä R0 tai vakiona 0

TTK-91 muistin osoitusmoodit (8)

ks. lista sivulla 50 [Häkk98]




```
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
```

Indeksointi (2)

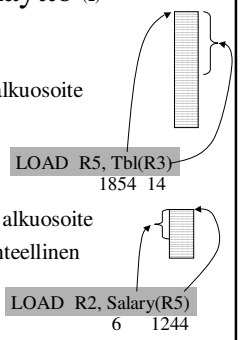
```
LOAD R4,=Tbl(R3)
```

- 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 ← 201
  - tyhjä: 1 kpl — R4 ← Mem[201] = 11
  - ”@”: 2 kpl — R4 ← Mem[Mem[201]] = Mem[11] = 300

pelkkä rekisterin nro @-merkin jälkeen ⇒ 1 kpl
STORE käsky ⇒ 1 kpl vähemmän noutoja ja yksi tallennus

Indeksoinnin käyttö (2)

- Taulukot
  - Vakio (symboli) taulukon alkuosoite
  - indeksirekisterissä indeksi
- Tietueet
  - indeksirekisterissä tietueen alkuosoite
  - vakiona tietueen kentän suhteellinen osoite tietueen sisällä



TTK-91 operaatiot

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

TTK-91 muistiinviittausoperaatiot (3)

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

TTK-91 I/O operaatiot

- IN — `IN R3, =KBD`
  - lue arvo (kokonaisluku) rekisteriin annetulta laitteelta
- OUT — `OUT R2, =CRT`
  - tulosta arvo (kokonaisluku) rekisteristä annetulle laitteelle
- Laitteet?
  - KBD - näppäimistö, stdin
  - CRT - näyttö, stdout
  - ei muita!

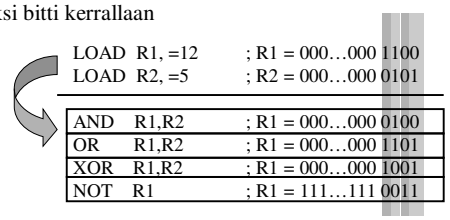
TTK-91 kokonaislukuoperaatiot

- 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
```

TTK-91 loogiset operaatiot (4)

- NOT, AND, OR, XOR
  - kaikille 32 bitille
  - yksi bitti kerrallaan



```
LOAD  R1, =12      ; R1 = 000…000 1100
LOAD  R2, =5       ; R2 = 000…000 0101
```

AND	R1,R2	; R1 = 000…000 0100
OR	R1,R2	; R1 = 000…000 1101
XOR	R1,R2	; R1 = 000…000 1001
NOT	R1	; R1 = 111…111 0011

TTK-91 bittien siirtokäskyt

- SHL, SHR
  - siirrä bittejä vasemmalle tai oikealle
  - täytä nollilla

```
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 (6)

- JUMP `JUMP Loop`
- COMP `COMP R3, =27` `COMP R2, X`
  - asettaa tilarekisteriin SR vertailun tuloksen: L, E tai G
- JLES, JEQU, JGRE, JNLE, JNEQU, JNGRE
  - perustuu tilarekisterin tietoon eli viimeksi suoritettuun COMP-käskyyn `JGRE Loop`
- JNEG, JZER, JPOS, JNNEG, JNZER, JNPOS
  - perustuu annetun rekisterin arvoon `JPOS R1, Loop`
- CALL, EXIT (käsitellään myöhemmin)
- SVC `SVC SP, =HALT ;ohjelman suoritus päättyy`

TTK-91 muut käskyt

- NOP `NOP`
  - Nop Operation, tyhjä käsky, älä tee mitään
  - varaa kuitenkin muistia yhden sanan (32 bittiä)
  - suoritetaan samoin kuin muutkin käskyt

TTK-91 assembler kääntäjän ohjauskäskyt (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, antaa sille alkuarvon ja antaa osoitteen symbolin arvoksi (symbolitauluun!)
  - esim. muuttujan tai ison vakion määrittely `LOAD R1, X`
- 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

TTK-91 symbolisia konekäskyesimerkkejä (10)

- 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

TTK-91 symbolisia konekäskyesimerkkejä (10)

- 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

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
```


TTK-91 symbolinen konekieliohjelma

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 rekisterissä?

        JUMP    Sum

Done    LOAD    R1, Summa       ; tulosta summa ja lopeta
        OUT     R1, =CRT
        SVC     SP, =HALT
```


KOKSI TTK-91 -koneen simulaattori (7)

- Toimii kuten oikea kone toimisi
- Graafinen käyttöliittymä
- I/O vain käyttöliittymän kautta
- Ohjelmien valinta (”lataus”), käännös ja suoritus
- Ohjelmien editointi

ks. sum.k91

  - myös mikä tahansa tekstieditori kelpaa!
- Käsky kerrallaan suoritus mahdollinen
- Käsky kerrallaan, kommentoinnin kera

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

-- Luennon 2 loppu --

Some typical 80x86 intructions and their function

Instuction	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 codes (flags) with EDX & 42
MOVSL	M[EDI]=M[ESI]; EDI=EDI+4; ESI=ESI+4

Fig. 3.32 [PaHe98]