## L3 - kombinatsioonskeemid

**Register** – Rühm ühise juhtimisega trigereid. Registri ülesandeks on ajutiselt numbrit salvestada, register asub CPU sees. On RAMist kiiremad. Saab info ALUst.

**ALU** – Arithmetic Logic Unit

**Kõige olulisemad kombinatsioonskeemid** – summaator (täissummaator) – ilma selleta on keeruline midagi luua. Olulisuselt teine on multipleksor.

**Multipleksori põhimõte** – nagu värav. 2-1 MUX ehks 2-ühele-multipleksor. Valikusignaal S väärtusele vastavalt suunatakse väravasse vastavalt A või B väärtus. Näiteks, S on 0, on väljund B, kui S on 1, siis väljund A.

**Mitu sisendit saab kontrollida 5 select signaaliga**

Ühe valikusignaaliga saab kontrollida 2 sisendit. 5 signaaliga saab 25

**Kas MUX näidet saab ka „odavamaks“ teha?**

Kui selline küsimus on loengus, siis on vastus „jah“. Näide on kulukas, sest kasutuses on AND ja OR on kallid. Odavamad on XOR (aga see pole siin oluline). Nende asemel NOR ja NAND on odavamad. NAND gate’id on riistvara mõttes oluliselt odavamad.

CMOS on inverteeritud tehnoloogia (inverteerib kõike).

**Mitu 2-1 MUXi on vaja, et realiseerida 8-1 MUX? Mitu select signaali on vaja?**

7 MUXi, sest saaks 8 sisendit algusesse ja edasi lähevad nagu turniiritabel.

3 select signaali on vaja. Esimeses rivis levib igale poole aga sellest väga aru ei saanud.

**ALU** – aritmeetika-loogikaplokk. Liitmine. Lahutamine, võrdlemine, nihutamine, AND, OR, NOT jne. Realiseerib selliseid asju.

**Mis operatsiooni saab riistvaras teha AND loogikatehtega?**Masking. Näiteks 8 bitisest registrist kätte saada ainult meid huvitavad bitid. Teed maski, kus meid huvitav osa on 1 ja mittehuvitav 0.

**Mitu MUXi on vaja 4 operatsiooni teostava ALU jaoks?**

Kolme

**Kuidas näidatakse summaatoris ületäitumist?**

Summa register on sama pikk nagu sisendite register. 11 + 10 = ~~1~~01. Tekkis ületäitumine. ALUl on ületäitumise näitamiseks ületäitumise lipp (overflow flag).

## L4 – järjestikskeemid – trigerid, register, loendurid

**Triger** – elementaarne mäluelement, salvestab infot 1-bitt. Enamasti 2 väljundit – otse ja pöördväärtus. Salvestavad kombinatsioonskeemides olekuid. Hulgaliselt tagasisidestusi ka, mis teeb trackimise keeruliseks :) Kõige lihtsam tõeväärtustabeli järgi vaadata.

JK trigeriga saab kõiki teisi simuleerida.

**Asünkroonne triger** – latch, reageerib sisendi muutusele. Kui sisend muutub, siis väljund muutub. Keelatud olek NOR elementidega on S1 R1, NAND elementidega keelatud S0 R0. Triger läheb lolliks, sest tagasisides tekib mingi jama (väljundid lähevad võrdseks) aga peab olema Q ja Q eitus.

Frondiga juhitav triger – flip-flop

**SR-triger** – Set Reset triger

*Eksamiks: Oluline on teha trigeritel vahet (et on asünkroonne ja sünkroonne, SRi puhul keelatud olekuid).*

**D-triger** (delay triger) – tööpõhimõttelt lihtsaim.

**JK-trigeri (jump-kill)** mõte on, et ei oleks keelatud olekut 1 1 või 00. Et ei tekiks sellist olukorda. Sisendid tulevad kombinatsioonskeemidest ja raske ennustada, mis sealt tuleb. Kui sisend on J1 K1, siis väljundit inverteeritakse.

Paneb nandid ritta ja teeb tagasiside.

**T-triger** – kõige lihtsam tõeväärtustabel.

**Mis on T-trigeri algolek?** Eraldi sisend selle jaoks. See, mis ta asünkroonsest asetussisendist alglaadimisel saab (nullimine või mingi kindel sisend).

**Registrid** – triger suudab 1 biti infot salvestada, muudab sisendile vastavalt oma väärtust ja kui vool ära läheb, siis ei hoia midagi. Rohkem bitte salvestada, siis teha hunnikust trigeritest registrid. Registrid on 99% juhtudest ühise sünkroniseerimisega. Ei taha, et trigerid registris suvaliselt väärtust muudaks. Eesmärk info säilitamine. Tahame algväärtustamist, et oleks võimalik registrisse laadida kindel signaal.

Nihkeregister – nihutamine on kasulik tehe, kuna tal on kindlad omadused. Vasakule, paremale, ringnihe.

**Nihe paremale** – arvu jagamine arvusüsteemi alusega (kahendsüsteemil 2). 1010 nihutamine paremale annab 0101, mis on 2x väiksem.
**Nihe vasakule** – arvu korrutamine alusega. See on odavam.

Nihkel tuleb sisse 0.

Paralleellaadimine on hea mõte. Muidu pikka registrisse panemisel (nt 128 bitti) läheb kaua.

**Loendurid** – sünkrosisend ehk loendussisend (clock). M-järgulisel loenduril on maksimaalselt 2m väljundkombinatsiooni.

**Sünkroonne loendur** – trigerid lülituvad ümber üheaegselt. Asünkroonne tähendab, et tekib läbi teiste trigerite viide.

## L6 – protsessor

**Von-Neumann** – mälu hoiab nii andmeid kui ka programme. Von Neumanni pudelikael on see, et käsud ja andmed on samal siinil.

**Protsessori osad** on juhtseade ja ALU.

**Mis kasu on käsustiku tundmisest?** Kui tunned käsustikku ja see sisaldab mingeid erilisi operatsioone, siis saame teha targema programmi, mis suudab neid ära kasutada. Mikrokontrollerite peal ujukoma arvutusi tehes näiteks. Võimsa prose puhul pole nagu oluline (seda ära ikka ei tapa).

Käsustiku **arhitektuur instruction set architecture** (**ISA**)

Kui arvuti käsu arhitektuur 16 bitti, käsukood 4 bitti. **Mitu käsku?** 16 käsku.
Kuidas eristada? Dekoodri abil.

**Mis käsku täidetakse kui protsessoril pole midagi teha?** Assembleri tasemel täidetakse käsku NOP (no operation).

**Käsuloendur** ehk käskude täitmise järjekorra järjehoidja protsessoris. Näitab järgmise käsu aadressi.

**Käsuregister protsessoris** on käsud kõik talletatud. Käsuga tuleb kaasa aadress või operand. Käsuregistrist saadetakse käsk dekoodrisse.

Adresseerimine – otsene ja vahetu.

**Milliseid andmeid võiks niimoodi (vahetu opereerimise tulemusena) edastada?** Operand manustatakse käsukoodi Konstante.

**Otsene adresseerimine** - käsuga antakse kaasa operandi aadress põhimälus, käsu juures on operandi aadress.

**Registermälu** on protessoris sees olev mälu, mis on väga kiire aga väga kallis, sest eeldab protsessoris lisapindala kasutamist ja meil on seda vähe. Registermälu 20 000 Mbaiti sekundis.

**TUUPIGE PÄHE:** Käsu täitmise tsükkel ehk Von-Neumanni tsükkel ehk fetch-decode-execute cycle –

1. Käsukoodi laadimine (fetch)
2. Käsuloenduri suurendamine PC = PC + 1 (öeldakse, mis on järgmise käsu aadress)
3. Käsukoodi dekodeerimine
4. Käsu täitmine

## L8 Arhitektuurid, Operatsioonautomaat, Juhtautomaat

**Siirdekäsu täitmine protsessoris** – käsukoodi laadimine (instruction fetch), käsukoodi dekodeerimine – käsu täitmine (instruction execute).

PC+1 – käsuloendurile+1

* PC Program counter
* MAR Memory address register (mäluaadressi register)
* MBR Memory buffer register (sellest võetakse käsukood)

**Millised registrid saavad anda sisendimälu aadressi registrile (MAR)?**

PC (programm counter ehk käsuloendur), IRp ehk Instruction register pointer või siis aadressi register. Kahe koha pealt saab MARile otsese adresseerimise puhul anda sisendit.

**Missugust digitaalloogika lülitust selleks kasutama peaks?**

Tahame pöörduda kahe erineva koha pealt ühele registrile ligi. Seega peame saama valida, kumb register annab oma selle küsimuse. Järelikult peaks kasutama multipleksorit 2-1.

**Juhtautomaat** (CU ehk Control unit) - käsu täitmise algoritmi riistvaraline realisatsioon loogikaskeemina. **Peamised automaadid** - **Mealy** automaat ja **Moore** automaat. Automaadi mõte on, et läheb tagasi algusesse kuni uuesti käivitatakse.

**Kombinatsioonskeem** - Skeemi väljundid sõltuvad ainult sisendist (L1)

**Mäluga skeem** – sisaldab mäluelemente (nt trigerid), tagasiside on vajalik. Väljund tõenäoliselt muutub aga sõltub.

## L9 - RISC ja CISC, Konveier, Hargnemiste ennustamine, Vahemälu

CISC käske realiseeriti mikroprogrammide abil.

**RISC** – lihtsad käsud, vähe käske, vähe adresseerimise viise, vähe käsuformaate. Käsusüsteemi tasandil RISC käsud lähevad riistvarasse täitmiseks (täidetakse ühe taktiga). Mahukas registermälu. Juhtautomaat jäiga loogikaga.

**CISC** – keerulised käsud, kasutab, kasutab mikroprogrammi. Käsusüsteemi tasandil käsud saadetakse kõigepealt mikroprogrammi (mis ütleb, mida on vaja selleks, et operatsioon läbi viia) ja alles siis riistvarasse täitmiseks.

Kui on siirdekäsk, siis vaadatakse, kas hargnemine toimub või ei. Hargnemise ennustamine parandab konveieri tööd.

* Fikseeritud hargnemiste ennustamine
* Staatiline hargnemiste ennustamine
* Dünaamiline hargnemiste ennustamine.

## L10 – Mälud. Konseptsioon, Hierarhia, Suvapöördus ja jadapöördusmälud, Säilivad ja mittesäilivad mälud

Mälu tasemed 0 (kõige lähemal) ja mida kaugemal (seda kaugemal ja aeglasem). Päris kaugel võib olla nt CD.

SAM – jadapöördusmälu. Serial Access memory.

**SRAM** – mälu elementideks on trigerid. Piisavalt kiire, et protsessoriga sammu pidada. Kasutatakse registermälu ja vahemälu realiseerimiseks.

## L11 – Mälud