**2016. zh kérdések A**

**CISC architektúra: Complex Instruction Set Computer Komplex utasításkészletű számítógép**

* nagyszámú utasítás-típus, címzési mód
* egy utasítással több feladat
* változó méretű utasítás formátum miatt lassabb (dekódoló, azonosítás, szükséges információ, feladat végrehajtás)
* korai gépek: egyszerű felépítés bonyolult nyelvezet 🡪 új nyelvek születtek (Fortran, Lisp, Pascal, C) – összetett problémákra
* komplexebb gépek: gyorsak, sokoldalúak
* ugyanaz a komplex program, függvény kevesebb utasítás sorozattal
* memória, regiszter alapú
* **ELŐNYE:** tömörség, kevés utasításból áll, kevesebb regiszterre van szükség
* **PÉLDÁK:** Intel 8080,8085, Zilog Z80, Intel 8086, DEC VAX

**Normalizált lebegőpontos rendszer:** rb=2 (rendszer alapszáma – mantisszára vonatkozik), re=2 (exponens alapja), e=6 (Excess-32), m=9 (jegyek száma – mantisszát reprezentálja), p=9 (tizedespont helye)

Rendszer jellemző paraméterei:

* mantissza értéke:
* VM(min)=1/ rb=0.5
* VM(max)=1.0 - rb-m
* VE(max)= re(e-1) - 1
* VE(min)= -( re(e-1) -1)
* lebegő pontos szám értéke:
* legális mantisszák száma:
* legális exponensek száma:
* ábrázolható értékek száma

**Szekvenciális hálózatok**

* van visszacsatolás
* korábbi állapot, és az aktuális külső bemenet függvényében határozzuk meg a kimeneteket
* digitális rendszer kontrollálható memóriával
* építőelemei: aszinkron tárolók (latch), szinkron tárolok és flip-flop, regiszterek (flip-flopok összekapcsolásából), számlálók, memóriák (ROM, RAM), időzítő-vezérlő egységek

**megadásának módjai:**

 **csoportosítás:**

* órajel nélküli aszinkron hálózatok (latch, aszinkron RS-tároló)
* órajellel vezérelt / időzített szinkron hálózatok (szinkron RS, MS flip-flop, JK flip-flop, D flip-flop)

**Hazárd:**

* késleltetés okozta nem kívánt kimenetek, állapotok
* kialakulhat: ha egy kapu kimenete a bemenetek változásához képest csak véges időn belül változik

propagation delay

interconnection delay: vezetéken lévő véges jelterjedés miatt

* fajtái: funkcionális/statikus

dinamikus

**Zérócímű gép használatával kiértékelés**

**Lebegőpontos kivonás esetén műveletvégzés képlete: A = B – C =
M B × r Eb - M C × r Ec = ( M B × r |Eb − Ec | - M C ) × r Ec**

**Kivonás blokkdiagramja**

**Blokkok funkciója**

* **Add/Substract egység:** aritmetikai rész
* **Exponent Compare**: két exponens összehasonlítása
* **SELECT/ALIGN egység**: előző eredménye kerül ide, helyre igazítja a megfelelő mantisszát
* **POST NORMALIZATION**: utólagos normalizálás megy végbe az összeadás/kivonás eredményén 🡪 kimenetre kerül

*0 és 1 közötti eredményt akarunk, így megfelelő bitpozícióval tud jobbra/balra shiftelni*

**Iteratív osztási művelet**

* **algoritmus menete:**
* számok normalizált lebegőpontosak, osztót osztandót törtkifejezésként
* osztót és osztandót ugyanazon **fK**értékkel szorozzuk (úgy választjuk, hogy nevező egységnyi legyen az iterációk elvégzése után)
* számláló értékéből kapjuk a **Q értéké**t
* **DS=1-x** ahol **x-et a DS határozza meg**
1. *f0 kiszámolása: f0 = 1+x = 1+(1- DS ) = 2 - DS*

*f 1= (1+x) (1+x)=* 1+(1−DS × fo )= 2− DS × fo

1. *fK -t megkapjuk, ha vesszük fK-1 és a DS (nevező) szorzatának 2’s komplemensét*
* **leállásának feltétele:** **fK közelítőleg 1**, akkor Q eredmény elegendően közel lesz a kívánt eredményhez

fix számú iterációs lépést végzünk (ROMot használunk)

* **működést biztosító áramkör rajza**



**4 x 4 bites direkt szorzó rendszer blokkdiagramja (Full Adder)**



**Szorzás sebessége, ha egy kapu késleltetése G**

**Szintézis:** A logikai szintézis **bemenete** valamilyen **hardver leíró nyelv** (pl. VHDL), amelynek segítségével a **rendelkezésre álló adatbázisok alapján a szöveges leírásból elektromos áramköri kapcsolási rajz generálható**.

M**agas szintű szintézis** (High-Level Synthesis):

* **szinkron digitális rendszerek logikai szint feletti automatikus tervezésé**t teszi lehetővé
* bemenete valamilyen **szekvenciális programnyelven** (pl. ANSI-C/C++) leírt **algoritmus**
* az eredményül kapott áramkör tartalmazza **az algoritmus utasításai** **által** definiált műveletek elvégzéséhez szükséges **funkcionális elemeket**
* a működés időzítéséhez szükséges **vezérlő egységet**

**2016. zh kérdések C**

**RISC architektúra** (Reduced Instruction Set Computer - Csökkentett utasítás készletű számítógép)

**Tulajdonságai**

* csak a kívánt alkalmazásra jellemző utasítástípusok
* felesleges utasítások nincsenek, utasítások egyszerűek
* processzorok **belsőfelépítése sokkal egyszerűbb, átláthatóbb** 🡪 hatékonyabb mikróarchitektúra kialakítása
* Load/Store és regiszter-regiszter típusú műveletek
* Kevés címzési mód
* fix utasításhossz
* Minden (vagy legalábbis a legtöbb) utasítás végrehajtása ugyanannyi ideig tart
* ELŐNY:
* egyszerűség 🡪 kevesebb tervezési hiba
* huzalozott vezérlés 🡪 magasabb órajel-frekvencia
* kisebb processzor 🡪 alacsony fogyasztás, kevés helyet foglal, jobb gyártási kihozatal
* HÁTRÁNY:
* elemi műveletek 🡪 nagyobb kód
* több regiszterre van szükség

**Példa**

* DEC Alpha 21164 EV5
* Motorola 88000 RISC rendszere
* Berkeley Egyetem RISC I rendszere

**Normalizált lebegőpontos rendszer – lásd feljebb**

**Kombinációs hálózat definíciója:** olyan áramkör, amely bemeneteket és kimeneteket tartalmaz. Ezek a pillanatnyi bemenetek egyértelműen meghatározzák a kimeneteket.

A kimenetet mindig csak a bemenetek függvényeként kapjuk.

Ezek esetén a NAND ill. NOR logikai alapkapuk univerzálisak, mivel az összes logikai fgv megvalósítható belőlük különböző kombinációval.

**Megadási módja:**

* Karnaugh tábla
* igazságtábla
* kapcsolási rajz
* függvény

**Zéró-című gép használatával kiértékelelés**

**Optimálisabb:** a hátulról – előre haladás optimálisabb, mert itt csak maximum 3 értéket kell egyszerre eltárolnunk.

**Lebegőpontos összeadás képlete: A = B + C = M B × r Eb + M C × r Ec = ( M B × r Eb − Ec + M C ) × r Ec**

**Blokkdiagramja**

**Blokkok funkciója:**

* **Add/Substract egység:** aritmetikai rész
* **Exponent Compare**: két exponens összehasonlítása
* **SELECT/ALIGN egység**: előző eredménye kerül ide, helyre igazítja a megfelelő mantisszát
* **POST NORMALIZATION**: utólagos normalizálás megy végbe az összeadás/kivonás eredményén 🡪 kimenetre kerül

*0 és 1 közötti eredményt akarunk, így megfelelő bitpozícióval tud jobbra/balra shiftelni*

**Direkt osztó blokkdiagram**

**Formula: *Dd = Q \* DS + R***

R maradék, Q hányados, Dd osztandó, DS osztó



Folyamatábra:

**Végrehajtás időszükséglete:** lassú folyamat

**Leállás feltétele:** *R < DS esetén leáll az osztás*

**4 x 4 bites direkt szorzó rendszer blokkdiagrammja Carry Save Adderrel**



**Sebessége, ha egy kapu késleltetése G:**

**Magas szintű szintézis (HLS) alapját adó sejtés**

**Elméleti alaptétele – Turning Church tézis:**

A szimbolikus számítások 3 fő ekvivalens reprezentánsai a következők:

* Turing gép (TM – hardver)
* µ-rekurzív függvény (algoritmus, program)
* CFG-környezetfüggetlen nyelvtan (OS)

Egy nyelv, amely leírja a teljes tervezési folyamatot, az egyes tervezési lépések jóságát méri:

 Nyelv HW