2018 digzer 12K B :

(1.) Magyarázza meg a regiszter ablakolás működését és alkalmazását.

processzoridő minimalizálására találták ki.
A nagyszámú regiszterhalmazból egyidőben csak adott, korlátozott számú regisztert használ a rendszer. Ezt a korlátozott számú regisztert egy ablakkal azonosítjuk, és elkülönítik a többi regisztertől.
Az ablak megváltozásakor egy pointer azonosítja, hogy az aktív regiszterek helyett új regiszterkészletet kell egy ablakba fogni.
Amikor az ablak átlapolódik a rutinok, alprogramok között, akkor a paraméterek átadását globális regisztereken keresztül oldja meg, amelyek az összes rutin által elérhetőek.
Az utasításkészletben 5 bittel azonosíthatjuk az utasításainál használni kívánt regisztert, eráltal 32 különböző regiszter (R0 - R31) címezhető meg (5 bit segítségével, mivel $2^5 = 32$).

Tehát vannak:
- közös (globális) regiszterek: R0 - R9, minden rutin által elérhetők.

- rutin specifikus regiszterek: R10 - R31 mely további 3 részből áll (a regiszterek között történhet átlapolódás):
  a) Alacsonyszintű regiszterek: R10 - R25
  b) Lokális regiszterek: R16 - R25
  c) Magasszintű regiszterek: R26 - R31

Ez az eljárás mindaddig jól működik, ameddig a paraméterek száma kisebb a regiszterek méreténél, mivel nem igényel memória-intenzív stack műveleteket.

(2.) Tekintsük a következő paraméterekkel adott 16 bites 2-es komplemens fixpontos rendszert: $p = 5$.

Mekkora a legkisebb (pozitív) ábrázolható szám?

Mekkora a legnagyobb ábrázolható szám?

Mekkora a legnegatívabb szám?

Mekkora a Δr ebben a rendszerben?

$V(\text{smallest absolute}) = 0000000000.00001 = 2^{-5} = \frac{1}{32} = 0{,}3125 \cdot 10^{-1}$

$V(\text{largest absolute}) = 0111111111.11111 = 1023{,}96875 \simeq 1024$

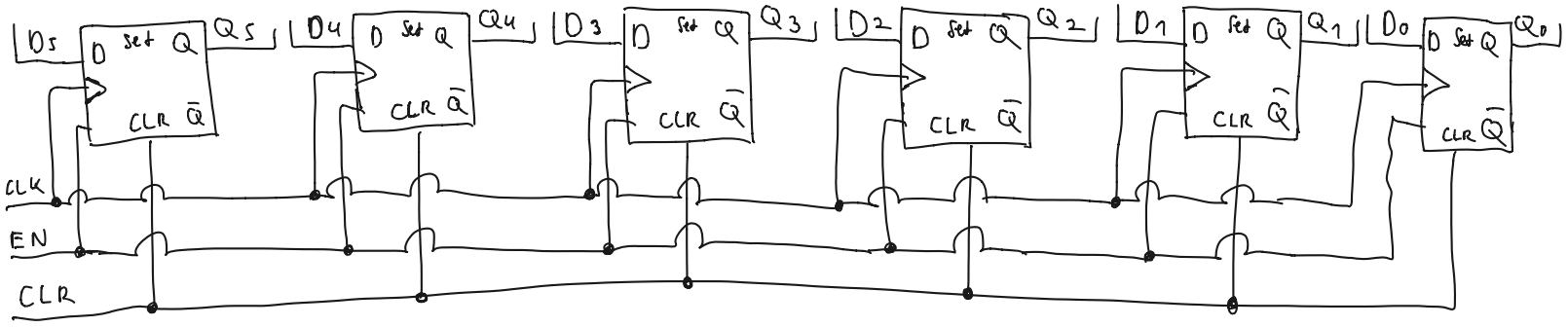
$V(\text{largest negative}) = 1000000000.00000 = -2^{10} \simeq -1024$

$\Delta r = 2^{-5} \simeq \frac{1}{32} = 0{,}3125 \cdot 10^{-1}$

2^{N-P-1}

$-\left(2^{N-P-1}\right)$

(3.) Adja meg egy 6 bites párhuzamosan írható és olvasható regiszter kapcsolását D tárolók felhasználásával.

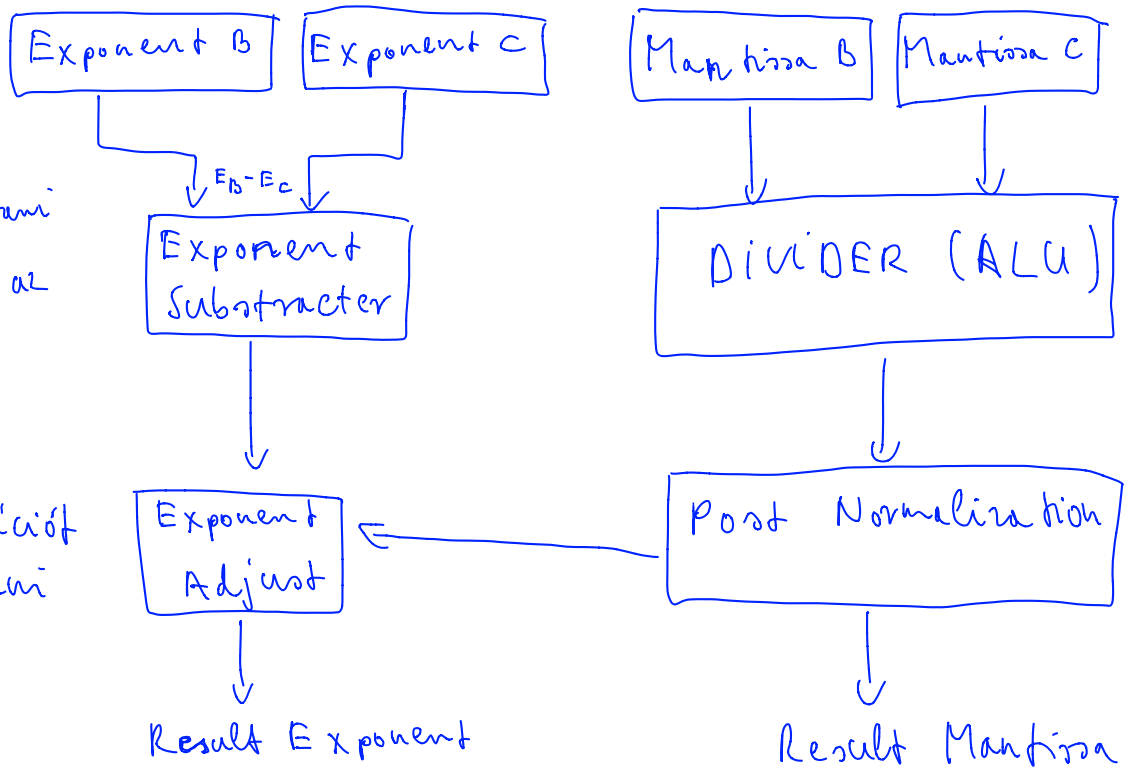


(4.) Adja meg a lebegőpontos osztó blokkdiagramját az egyes blokkok funkciójával együtt.

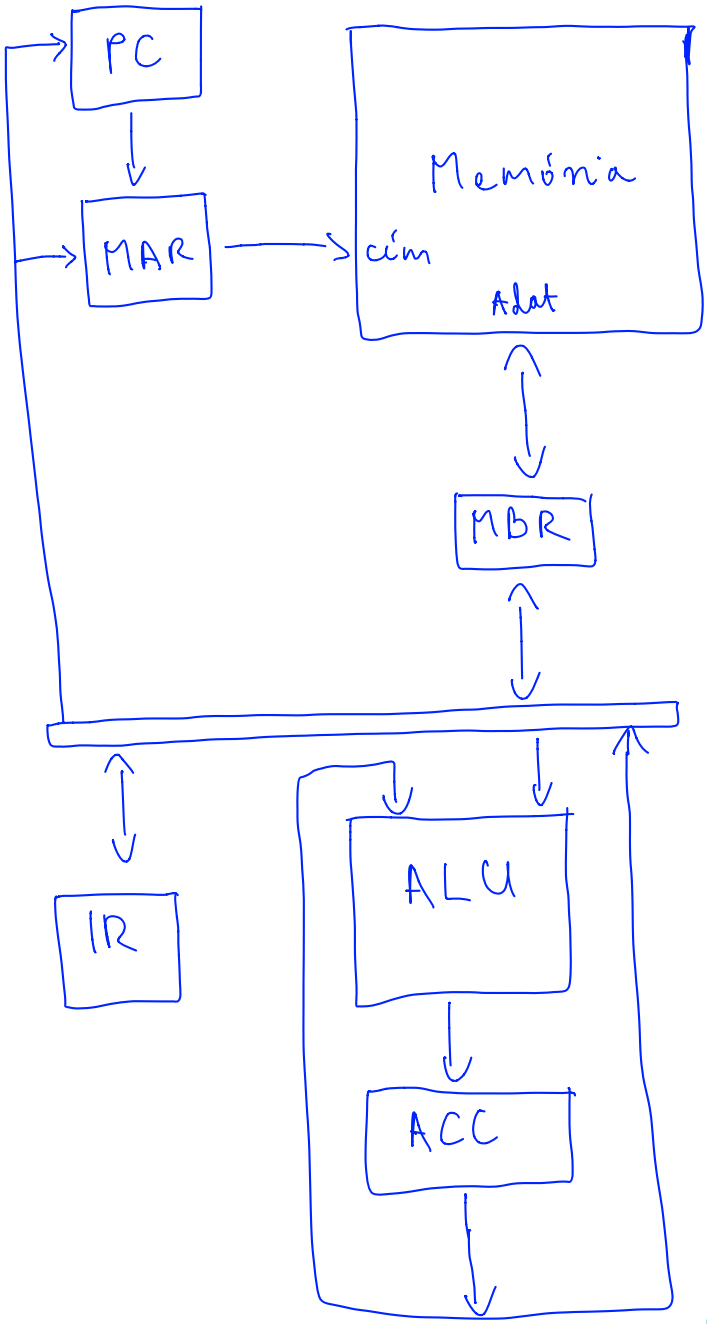
Művelet: $A = B/C = M_B \times r^{E_B} / M_C \times r^{E_C} = (M_B/M_C) \times r^{E_B - E_C}$

A: hányados
B: osztandó
C: osztó

* Könnyű végrehajtani
* Nincs szükség az operandusok beállítására
* Minimális post-normalizációt kell csak végezni
* Osztás



(5.) Rajzolja fel az egycímű számítógép blokkdiagrammját! Definiálja az egyes blokkok funkcióját is.



* MAR: Memory Address Register.
(Memória-cím Regiszter):
a memóriából érkező, vagy a memóriában lévő információ helyét azonosítja adott memóriacím alapján.

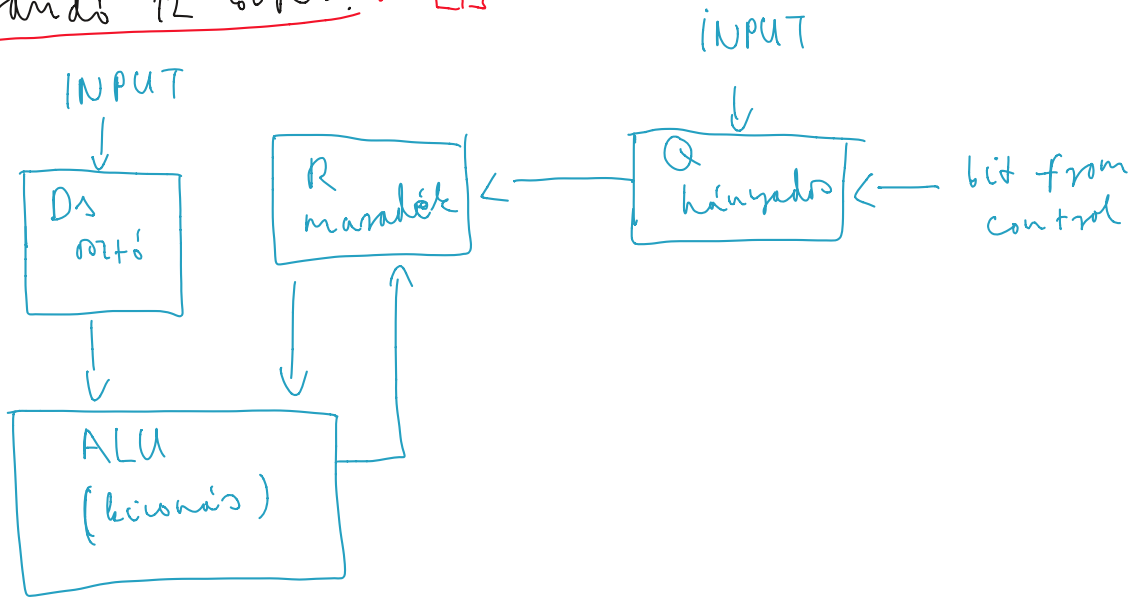
* MBR: Memory Buffer Register.
(Memória Puffer Regiszter):
tárolja a memóriába beírt, ill. az onnan érkező információt.

* PC: Program Counter
(programszámláló):
a soron következő (végrehajtandó) utasítás helyét azonosítja.

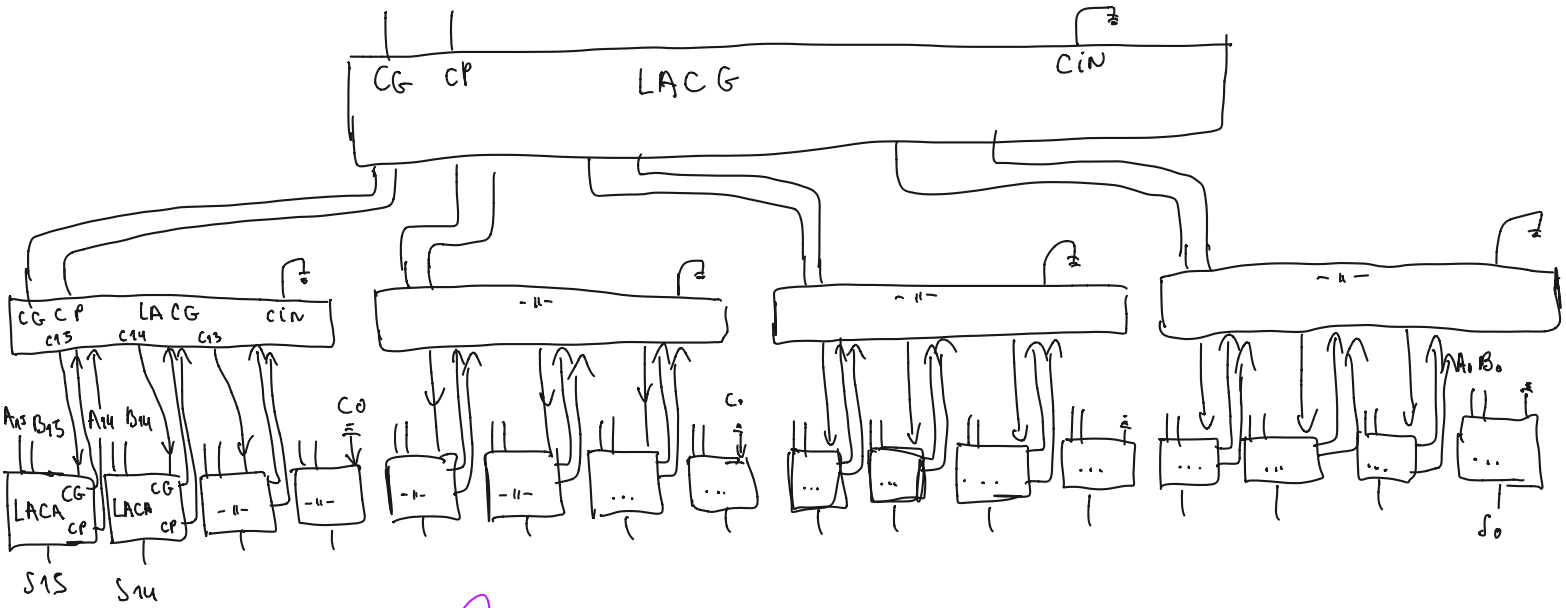
* IR: Instruction Regiszter
(utasítás Regiszter):
tárolja az éppen végrehajtás alatt álló utasítást.

* ACC: Accumulator (tárolóregiszter): eredmény ideiglenes tárolására használjuk (összes adatkezeléshez tartozó utasítás tárolása).

6. Adja meg egy direkt osztó blokkdiagramját és adja a végrehajtás időszükségletét, ha az osztó 6 bites és az osztandó 12 bites. * (!)

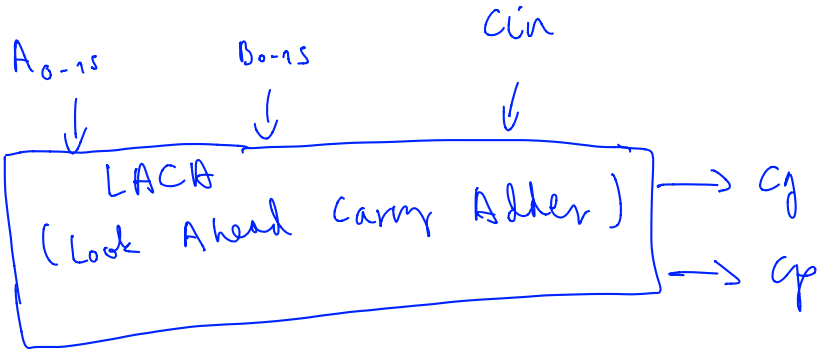


(7.) Adja meg egy 16 bites LACA (Look-ahead Carry Adder) összeadó blokkdiagramját, és adja meg a 16 bites összeadás időszükségletét (4 bites LACG generátort feltételezve)



Kapcsolás

Blokkdiagram:



időszükséglet: $C_g = AB$ $C_p = A + B$

$$T_{LACA} = 2 + 4 \times \left(\left[\underbrace{\log_4(16)}_{2} \right] - 1 \right) = 6$$

b -> Hány bites a LACG

N -> Hány bitet akarok összeadni

8. Definiálja az allokáció fogalmát (allocation) magas-szintű szintézis esetén!

"felosztás"

Amikor a feladatot többféle FU (funkcionális egység) közül kell megoldani, ez a hozzárendelés választja ki a legmegfelelőbb FU-t az elvégzéséhez.

A teljes kezdő adatút generálása és optimalizálása

9. Adott: memória hozzáférés ideje 20 ns, regiszterből regiszterbe másolás 4 ns, kivonás 6 ns. Adja meg a ADD3 X, Y, Z RTL leírását és időigényét.

3 című gép
direkt címzés

Fetch: (regiszterek feltöltése, utasításhívások):

PC->MAR	4 ns
M[MAR]->MBR	20 ns
PC + I_len -> PC	4 ns
MBR -> IR	4 ns
	Σ 32 ns

Decode: (a dekódolást általában 0 idejűnek feltételezzük)

Execute: (végrehajtás)

PC -> MAR	4 ns
PC+XA_len -> PC	4 ns
M[MAR] -> MBR	20 ns
MBR -> MAR	4 ns
M[MAR] -> MBR	20 ns
MBR -> T1	4 ns

Ha indirekt + 20 ns X címének a címe

PC -> MAR	4 ns
PC+YA_len -> PC	4 ns
M[MAR] -> MBR	20 ns
MBR -> MAR	4 ns
M[MAR] -> MBR	20 ns
MBR -> T2	4 ns

Ha indirekt + 20 ns Y címének a címe

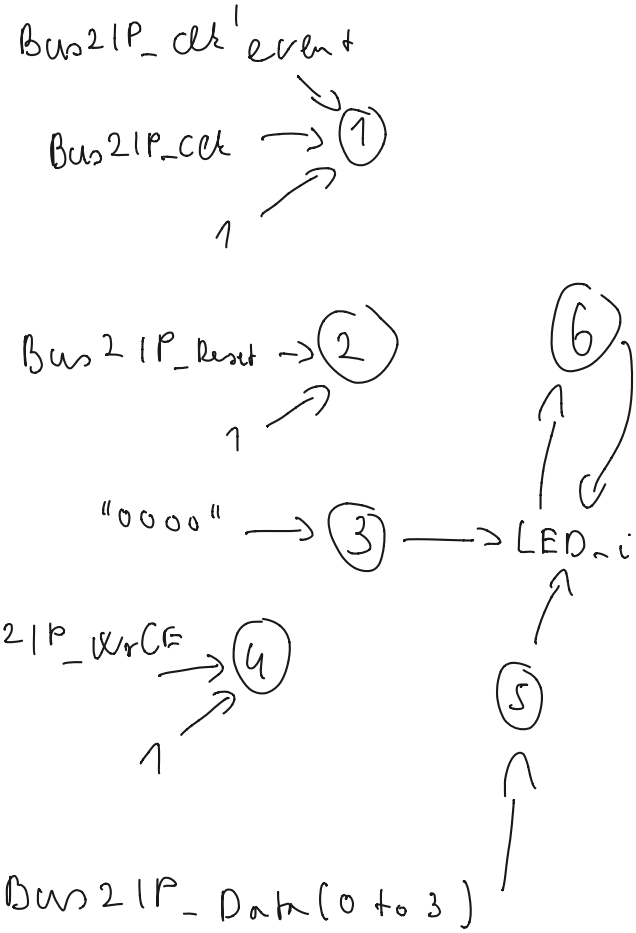
PC -> MAR	4 ns
PC+ZA_len -> PC	4 ns
M[MAR] -> MBR	20 ns
MBR -> MAR	4 ns
T1 - T2 -> MBR	6 ns
MBR -> M[MAR]	20 ns

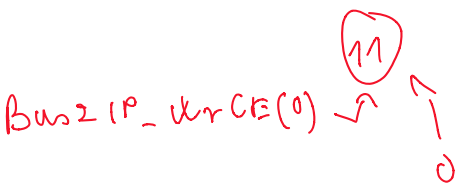
44 + 120 + 6 => ⌈ 170 ns

10.

10. Adja meg az alábbi VHDL szubrutin DFG (adatfolyam) grafját:

```
LED_PROC : process (Bus2IP_Clk) is
begin
  if Bus2IP_Clk'event and Bus2IP_Clk='1' then 1.
    if Bus2IP_Reset='1' then 2
      LED_i<="0000"; 3.
    else 10
      if Bus2IP_WrCE(0)='1' then 4.
        LED_i<=Bus2IP_Data(0 to 3); 5.
      else 11
        LED_i<=LED_i; 6
      end if; 7
    end if; 8
  end if; 9
end process LED_PROC;
```





11. Sorolja fel a kombinációs hálózatok leírási módjait.

* Algebrai alak
* Igazságtábla
* Algebrai normál alakok
* Karnaugh tábla
* (Quine - Mc Cluskey Módszer ~ Igény szerint)