NÉV:..... Neptun kód:..……

A feladatokat önállóan, meg nem engedett segédeszközök használata nélkül oldottam meg:

Olvasható aláírás:.....……

Kedves Kolléga! ***A kitöltést a név és aláírás rovatokkal kezdje!*** Az alábbi kérdésekre a válaszokat - ahol lehet - mindig a feladatlapon adja meg! A feladatok megoldása során a részletes kidolgozást nagyfeladatonként, ha szükséges külön papíron végezze, (egyértelműen jelölje, hogy melyik lap melyik feladathoz tartozik, a papírra már a kezdetkor írja rá a nevét és Neptun kódját) és ezeket a papírokat is adja be a dolgozatával! A kérdésekre a táblázatok vagy a pontozott vonalak értelemszerű kitöltésével válaszoljon, hacsak külön másként nem kérjük. ***Mindenütt a legegyszerűbb megoldás éri a legtöbb pontot.*** Jó munkát!

E:
F1:
F2:
F3:
Σ
IMSC2:
IMSC3:

Ellenőrző kérdések (27p)

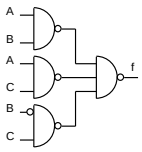
E1. (3p) Konvertálja az alábbi számokat a kért formátumra!

kiinduló formátum	konvertálandó szám	kért formátum	konvertált szám
2 digites decimális	59	8 biten BCD	
6 bites 2-es komplemens	100001	8 bites 2-es komplemens	
decimális	6.25	5 bites fix pontos előjel nélküli bináris, 2 db kettedes tört jeggyel	

E2. (1p) Egyszerűsítse az alábbi Boole kifejezést!

/A + AB + BC = ...

E3. (4p) Töltse ki az alábbi kapcsolás rajzzal megadott f logikai függvény igazságtábláját!



A	B	C	**f**
0	0	0	
0	0	1	
0	1	0	
0	1	1	

A	B	C	**f**
1	0	0	
1	0	1	
1	1	0	
1	1	1	

E4. (3p) Igazságtáblájával adott az f logikai függvény.

a. Adja meg a függvény diszjuktív normál alakját (DNF tehát ne egyszerűsítsen)!

a	b	c	**f**
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1

a	b	c	**f**
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

f_{DNF}= .....

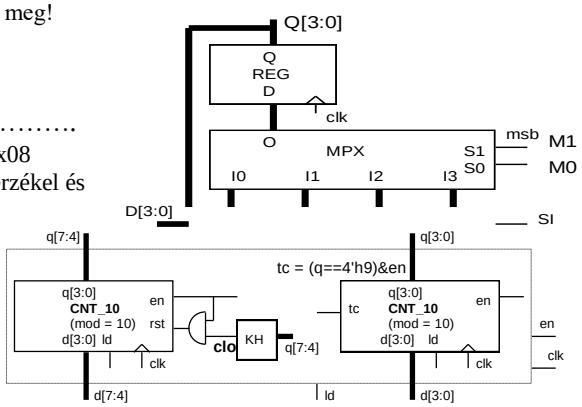
b. Milyen ismert funkcionális elem mely kimenetét állítja elő az f függvény?

.....

E5. (3p) A mellékelt blokkvázlatról hiányoznak a I0, I1, I2, I3 jelek. Adja meg ezeket úgy, hogy 4 bites törölhető, tölthető, engedélyezhető, balra shiftelő shiftregisztert valósítson meg!

M1M0:	00	01	10	11
funkció	tart	tölt	balra shiftel	töröl

I0:……. I1:...…… I2:……. I3: ...…….



E6. (2p) A MiniRISC CPU Éppen a 0x10 címen található **jsr** 0x08 utasítást hajtja végre. Az utasítás execute fázisában **IRQ =1**-et érzékel és elfogadja az interrupt kérést.

a. Milyen címről olvassa a következő utasítást? ……

b. Milyen címre ugrik az **rti** utasítás hatására?.....

E7. (3p) Egészítse ki a jobb oldali rajzot a következők szerint.

a. Csökkentse a bal oldali számláló modulusát 6-ra! Adja meg Verilog nyelven az ehhez szükséges clo kimenetet előállító logiát!

assign clo = ……..

b. Egészítse ki a rajzot úgy, hogy a két számláló együtt egy 60-as modulusú (0,1…59, 0…) engedélyezhető, tölthető számlálót valósítson meg!

E8. (3p) Válaszoljon a következő MiniRISC GPIO perifériával kapcsolatos kérdésekre! A periféria ADO regiszterébe 0x00-t írunk, a periféria lábain tápra húzó ellenállások vannak elhelyezve, kívülről semmi nem kapcsolódik rá. (Többlet információk a GPIO perifériáról az F3 feladat elején található.)

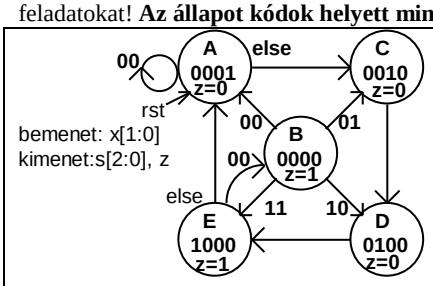
- **a.** A periféria bitjeit kimenetként állítva milyen logikai értéket lehet olvasni az ADI regiszteréből? …….
- **b.** A periféria bitjeit bemenetként állítva milyen logikai értéket lehet olvasni az ADI regiszteréből? …….
- **c.** Hány MiniRISC *órajel* alatt lehet a GPIO kimenetét megváltoztatni? ……

E9. (5p) Mely állítások igazak és melyek hamisak? Jelölje ***+-al az igaz, −-al a hamis*** állításokat!

1.	Csak AND kapukkal, OR kapukkal és inverterekkel minden logikai függvény előállítható.	
2.	LOAD-STORE architektúrájú processzor esetén egy művelet eredménye csak regiszterben keletkezhet.	
3.	Két UART összekapcsolásánál az azonos elnevezésű jeleket kell egymással összekötni.	
4.	A 2 regiszter címes processzor architektúra több bites utasítás kódot igényel, mint a 3 regiszer címes.	
5.	Az egyszintű interrupt rendszerekben egy interrupt megszakíthatja egy másik végrehajtását.	

Feladatok:

F1. (13p) Adott egy FSM az alábbi ***kódolt állapotgráfjával.*** A kimenete az állapotkód és **z**. Végezze el az alábbi feladatokat! **Az állapot kódok helyett mindenütt állapot neveket használjon!**



a. (3p) Adja meg az állapotregiszter Verilog leírását!

```
//A konstans és változó deklarációk
localparam [3:0] A = 3'b0001, B = 3'b0000,
    C = 3'b0010, D = 3'b0100, E = 3'b1000;
reg [3:0] s, next_s;
wire clk, rst;
wire [1:0] x;
//Állapotregiszter
always @ ( ……)

   if (rst)  s <= ……;

   else      s <= ……;
```

b. (7p) Következő állapot logika

```
//Next_state logika
always @ (.................)
case (s)
    A:if(~x[1]&~x[0]) next_s <= ……;

      else            next_s <= ……;
    B:  case(……)
        2'b00: next_s <= ……;

        2'b01: next_s <= ……;

        2'b10: next_s <= ……;

        2'b11: next_s <= ……;
        endcase
    C:                next_s <= ……;

    D:                next_s <= ……;

    E:if……)           next_s <= ……;

      else            next_s <= ……;

   default:           next_s <= ……;
endcase
```

c. (1p) Milyen modell szerint működik? (Húzza alá a megfelelőt) Mealy Moore

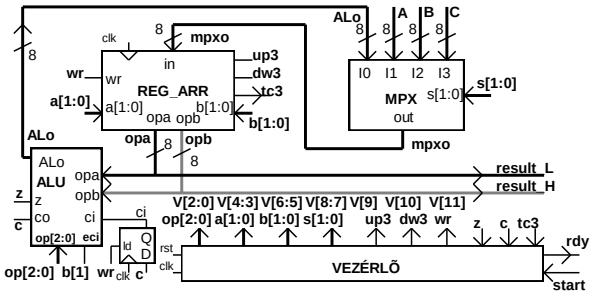
d. (2p) Adja meg a z kimenet logikai függvényét Verilog-ban!

A legegyszerűbb SOP alakban: **assign z =** ……..…

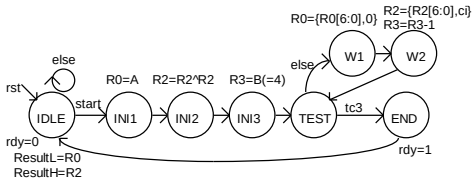
Ha csak az s-t, az állapot neveket, az == operátort és | műveletet használhatja:

assign z = ……..

F2. (18p) Funkcionális blokkvázlatával (adatstruktúra + vezérlő) adott egy a bemenő adatokkal többféle művelet elvégzésére alkalmas számító modul. Az adatstruktúra **3 db 8 bites adat bemenettel** rendelkezik: **A, B, C** (előjel nélküli számok). Az aritmetikai logikai egység **(ALU**) 8 műveletet tud elvégezni az **opa** és **opb** operandusokkal az alább megadott táblázat szerint. Az **elvégzendő művelet** az ALU **op[2:0]** bemenetén állítható be. **Shiftelés esetén a belépő bit 0, ha eci = 0, ci, ha eci =1. Összeadás/kivonás esetén a ci-t csak akkor veszi figyelembe, ha eci = 1**. A **művelet eredményét** az **ALo** kimenet adja. Az ALU **z** kimenet 1 értéke jelzi, ha a **művelet eredménye 0**. A **c** kimenet **összeadás esetén a túlcsordulást, kivonás esetén a negatív eredményt, shiftelés esetén a kilépő bit** értékét jelzi. A **REG_ARR egy 4 regisztert tartalmazó regiszter tömb.** Ennek **opa kimenetén az a[1:0]-val kiválasztott sorszámú regiszter tartalma** jelenik meg, **ha a[1:0]=i, akkor Ri**. Az **adat beírását wr=1 engedélyezi és az in bementére kiválasztott adat** (mpxo) **az a[1:0]-val kiválasztott regiszterbe** íródik az órajel felfutó élére. A **wr jel** ezen kívül **az ALU c kimenetét beírja az ALU ci bementére kapcsolt 1 bites regiszterbe**. Az **opb kimenetén a b[1:0]-val kiválasztott regiszter tartalma** jelenik meg, **ha b[1:0]=i, akkor Ri**. Az ALU **eci** bemenetére **b[1]** van kötve, ezért a **ci bemenetét csak akkor veszi figyelembe az ALU, ha opb-n R2 vagy R3 van kiválasztva**. A **REG_ARR bemenetére kerülő adat** (mpxo) **az MPX multiplexer s[1:0] bemenetével** választható ki. **A regisztertömb R3 regisztere speciális, mert fel/le számlálóként is funkcionál.** Normál regiszterként írható (a[1:0]=2'b11 és wr = 1 esetén beíródik R3-ba az MPX-el kiválasztott érték) és normál regiszterként olvasható**.** Azonban, ha éppen nem írják, akkor **up3,dw3 = 0,1 esetén lefele számol (R3=R3-1), up3,dw3 = 1,0 esetén pedig felfele számol (R3=R3+1),** ha megjön a clk aktív éle**. Az írás erősebb, mint a számlálás engedélyezés. Az R3 regiszter fel vagy le számláltatásával egyidőben bármely más regiszterbe lehetséges írni.** A **tc3** kimenet akkor jelez, ha **R3==0**. A vezérlőt egy kívülről jövő, 1 órajel hosszú **start** parancs indítja. A vezérlő az adatstruktúrát a **V[11:0] vezérlő jelekkel** működteti. A működése közben figyelni képes az adatstruktúrából jövő **z, c** és **tc** feltétel jeleket. Egy számítási feladat elkészültét 1 órajel hosszú **rdy** státus jellel kell jeleznie. Az eredménynek meg kell maradnia a következő start jelig. **Az aktuális művelet szempontjából érdektelen vezérlő jelek értékét 0-nak kell választani**. Egy maximum 8 bites végeredményt a regiszter tömb opa kimenetén (**result_L**) kell megjeleníteni. Ha a végeredmény megjelenítéséhez 8-nál több bit kell, akkor a többi bitet az opb kimeneten (**result_H**) kell megjeleníteni.



a. (8p) Az alábbi Moore jellegű HLSM állapot diagram által leírt feladatot a fenti adatstruktúrával kell megvalósítani. (R0-at szorozza 16-al, eredmény:R2R0-ban)



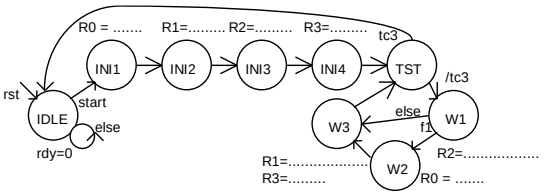
ALU funkció vezérlés: op[2:0]	ALU out (Alo)	z =1, ha	c = 1, ha
000	opa + opb + ci&eci	Alo==0	átvitel van
001	opa – opb – ci&eci	Alo==0	a–b < 0
010	{opa[6:0], ci&eci}	Alo==0	opa[7]
011	{ci&eci, opa[7:1] }	Alo==0	opa[0]
100	opa & opb	Alo==0	Soha (c=0)
101	opa | opb	Alo==0	Soha (c=0)
110	opa ^ opb	Alo==0	Soha (c=0)
111	opb	Alo==0	Soha (c=0)

Adja meg a ***vezérlő állapotgráfjának megadott állapotaiban kiadandó vezérlőjeleket a V[11:0] bitek értékének (0, 1) alábbi táblázatba való beírásával!*** (Segítségképp néhány értéket megadtunk. A rdy-t most nem kérjük.)

	wr	dw3	up3	s[1:0]	b[1:0]	a[1:0]	op[2:0]
V[11:0]	**11**	**10**	**9**	**8 7**	**6 5**	**4 3**	**2 1 0**
IDLE				0 0			0 0 0
INI1					0 0		0 0 0
INI2							
INI3					0 0		0 0 0
TEST				0 0	0 0	0 0	0 0 0
W1							
W2							
END				0 0	0 0	0 0	0 0 0

b. (1p) Milyen értékű lesz R2 az INI3 állapotban ?
……..

c. (7p) A számító modullal két 4 bites szám szorzatát kell kiszámítani, **ResultL = A*B**.
A megvalósítást: ***Kezdéskor a szorzandót R1, a szorzót, R2 regiszterbe tesszük, az eredmény regisztert 0-ázzuk. A szorzást ciklusba szervezzük Ciklusszámlálóként a speciálisan számlálóként is használható R3 regisztert használjuk. A szorzást LSB-vel kezdjük. A szorzó LSB-jét megjelenítjük c-ben. Ha c=1, akkor az eredményként használt regiszterhez hozzáadjuk a szorzandó aktuális 2 hatványát, ha c=0, nem adjuk hozzá. Mindkét eset után előállítjuk a szorzó következő 2 hatványát és ezzel egyidőben csökkentjük a ciklusszámlálót. Ezután visszamegyünk a ciklus elejére, ahol ellenőrizzük a ciklus számlálót. Rendelje a számítás elvégzéséhez szükséges műveleteket a Moore jellegű HLSM állapotaihoz.*** Az alább felsorolt műveletekből válasszon (szükségtelenek is vannak közöttük). ***Írja az elvégzendő művelet(ek) sorszámát a megfelelő állapot neve mellé!*** Nem biztos, hogy van elvégzendő művelet, ilyenkor írjon x-et! (A HLSM gráf segítő információkat tartalmaz. A gráf pontozott részeit nem kell kitölteni, de az is segítő információ.) ***A többlet és hiányzó művelet sorszámokért pontlevonás jár!***



1: R0 = {R0[6:0],0} **2:** R0 = R0^R0 **3:** R0 ={0, R0[7:1]}
4: R0 = R0+R1 **5:** R1 = A **6:** R1 = B **7:** R1 = C
8: R1 = R0+R1 **9:** R1 = {R1[6:0],0} R2 = A **10:** R2 = B
11: R2 = C **12:** R2 = {0, R2[7:1]} **13:** R3 = R3+1
14: R3 = R3-1, **15:** R3 = R3+R1 **16:** R3=C **17:** rdy = 1

IDLE:..….. INI1: …...… INI2:..…… INI3: ……

INI4: …….TST: ……. W1:…… W2: ……..

W3:……..

d. (1p) ***Adja meg a kért feltéte(lek)et a blokkvázlaton szereplő feltétel jel(ek) felhasználásával.*** (A parancs és feltétel bitek negáltját is fel lehet használni.)

f1: ….…….

e. (1p) Adja meg az adatstruktúra C bemenetének értékét! (Az algoritmusban fel kell használni.)

C = ……

f. IMSC_F2 (5p) Készítsen HLSM diagram részletet a feladatban szereplő adatstruktúrához, a következő feladatra: Dekrementálja a R2R0-ban levő 16 bites számot. Röviden magyarázza el szövegesen is a megoldását!

f. folytatása:

F3. (17p) A MiniRISC **GPIO A** egysége segítségével kell megoldania egy feladatot.
A GPIO A adatregiszterébe (**ADO**) kell írni a kimenetek értékét. Az irányregiszterével (**ADR**) állítható be bitenként a port iránya. Amely bitekre 1-et írunk, azok kimenetek lesznek, vagyis az I/O lábakon megjelenik az adatregiszterbe írt bitek értéke. (ADR az adatregiszter kimeneti bitjeire kapcsolódó 3 állapotú meghajtókat engedélyezi vagy tiltja.) Az ADO és ADR regiszterek tartalma visszaolvasható. Az I/O lábak aktuális értékét az **ADI** regiszterből lehet beolvasni. Első feladatként a **GPIO A** periféria hardver egység egy részét kell megtervezni (újratervezni) Verilog nyelven. **A MiniRISC busz felhasználható jelei: (A[7:0], Din[7:0], Dou[7:0], RD, WR, IRQ, clk, rst).** A GPIO A periféria báziscíme **0xA0.**

A GPIO A periféria programozói felülete:

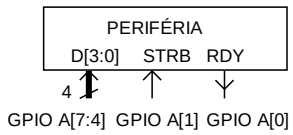
Funkció	Cím	D[7:0]	olvasható/írható
Kimeneti adat regiszter ADO	Báziscím + 0	ADO[7:0]	W/R (***ADO_w, ADO_r***)
Adat az I/O lábakon ADI	Báziscím + 1	ADI[7:0]	R (***ADI_r***)
Irány regiszter ADR (0: be, 1: ki)	Báziscím + 2	ADR[7:0]	W/R(***ADR_w, ADR_r***)

a. (5p) Tervezze meg Verilog nyelven a szükséges engedélyező jeleket: ADO regiszter írás engedélyező ***ADO_w***, olvasás engedélyező **ADO_*r*,** ADI regiszter olvasás engedélyező **ADI_*r*,** ADR regiszter írás engedélyező **ADR_*w***, olvasás engedélyező **ADR_*r***!

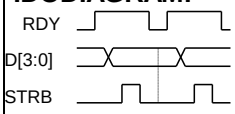
b.
```
parameter base_addr = ……;     // A periféria regiszter címtartomány kezdőcíme hexadecimálisan
assign psel = ……;    // Aktív,ha a periféria címtartományához fordul a proc.
assign ADO_w = ……;
assign ADO_r = ……;
assign ADI_r = ……;
assign ADR_w = ……;
assign ADR_r = ……;
```

b. (3p) Definiálja Verilog nyelven az GPIO A visszaolvasható regisztereinek olvasásához szükséges logikát az alábbiak szerint! Elkezdtük, folytassa!

```
reg [7:0] ADO, ADI, ADR;
reg [7:0] Din;
always @ (.....................)
   case ({ADR_r, ADI_r, ADO_r})
        3'b100: ……<= ……;
        3'b010: ……<= ……;
        3'b001: ……<= ……;
        default: ……<= ……;
   endcase
```



IDŐDIAGRAM:



A c pont után következő feladatok (d, e)

Egy periféria **4 bites** adatokat vár. Az **RDY** jellel jelzi, ha **új adat írható bele**. Ekkor a **D[3:0]** bementére kell adni a **beírandó adatot** és egy **STRB impulzust kell adni**, miközben a D[3:0] nem változik. A periféria a GPIO_A perifériával van összekötve, a fenti rajz szerint. A kommunikáció időzítését alatta mutatjuk. **Programozott bit billegtetéssel** (bit-banging) kell megoldania a feladatot. Feltételezzük, hogy az időzítési követelmények automatikusan telesülnek a MiniRISC-kel történő kezelés esetén.

c. (2p) Tervezze meg a 8 bites adat regisztert (ADO)! A regisztert a rst törli. Ha a processzor a címére ír, beleíródik az adat.

```
reg[7:0] ADO;
always @ (.............................)
if (……)          ADO <= ……
   else
       if (……) ADO <= ……
```

További feladatok:

e. (4p) Írja meg a PER_wr szubrutint, mely az r10 regiszterben megadott adat alsó és felső 4 bitjét ebben a sorrendben kiírja a perifériába **az idődiagrammal megadott módon**! A feladatban nem használt GPIO bitekre az adat regiszterben mindig 0-át írjon!

```
PER_wr: mov r11, …….   ; PIN. beolvasása
        ……..……..……   ; RDY bit ellenőrzése
        ……..……..……   ; vissza, amíg nem 1
        mov  r11,……..  ; adat másolása
        ……...r11,……..  ; adat alsó 4 bit kimaszkolása
        ……..……..……   ; adat alsó 4 bit a felső 4 bitre
        ……..……..……   ; adat kiírása a GPIO-ba (STRB:0)
        …….. r11,……   ; STRB bit 1-be állítása
        ……..……..……   ; adat kiírása a GPIO-ba (STRB:1)
        …….. r11,……   ; STRB bit 0-ba állítása
        ……..……..……   ; adat kiírása a GPIO-ba (STRB:0)
wait:   mov r11,……..   ; PIN. beolvasása
        ……..……..……   ; RDY bit ellenőrzése
        ……..……..……   ; vissza, amíg nem 1
        ……. r10,……..  ; adat felső 4 bit kimaszkolása
        ……..……..……   ; adat kiírása a GPIO-ba (STRB:0)
        …….. r10,……   ; STRB bit 1-be állítása
        ……..……..……   ; adat kiírása a GPIO-ba (STRB:1)
        …….. r10,……   ; STRB bit 0-ba állítása
        ……..……..……   ; adat kiírása a GPIO-ba (STRB:0)
        ……..……..……   ; visszatérés a szubrutinból
```

d. (3p) Adja meg a periféria kezeléséhez szükséges assembly nyelvű definíciókat, majd programozza fel a GPIO_A[7:4]-et és GPIO A[1]-et kimeneti portnak, és írja ki ***az idődiagram szerinti kezdőértéket (D[3:0] kezdetben legyen 0)***. Az ADR, ADO regiszterek címei már definiált konstansok, azokat nem kell definiálnia..

```
DEF RDY    0b00000100 ;RDY mask
DEF STRB   0b……..  ;STRB mask
DEF NSTRB  0b……. ;STRB mask negáltja
             CODE
Itt még ne használja fenti DEF konstansokat, a
kódban adja meg a megfelelőt hexadecimálisan!
RST:  mov r0, #0x00  ;GPIO_A kezdőérték
      ……..  ;kezdőérték az adat reg.-be
      mov r0, ……..  ;GPIO_A [7:4] és [1] irány
      ……..  ;kimenetnek állítása
```

IMSC_F3 (6p) Írja meg külön lapon az **e**-pontbeli szubrutin olyan változatát, amely csak az r10-ben megkapott adat felső 4 bitjét írja ki a perifériára, de **a GPIO_A adatregiszterének nem használt bitjeit nem változtatja meg**.

Rendelkezésre álló idő: 100 perc **Összes pont: 75**