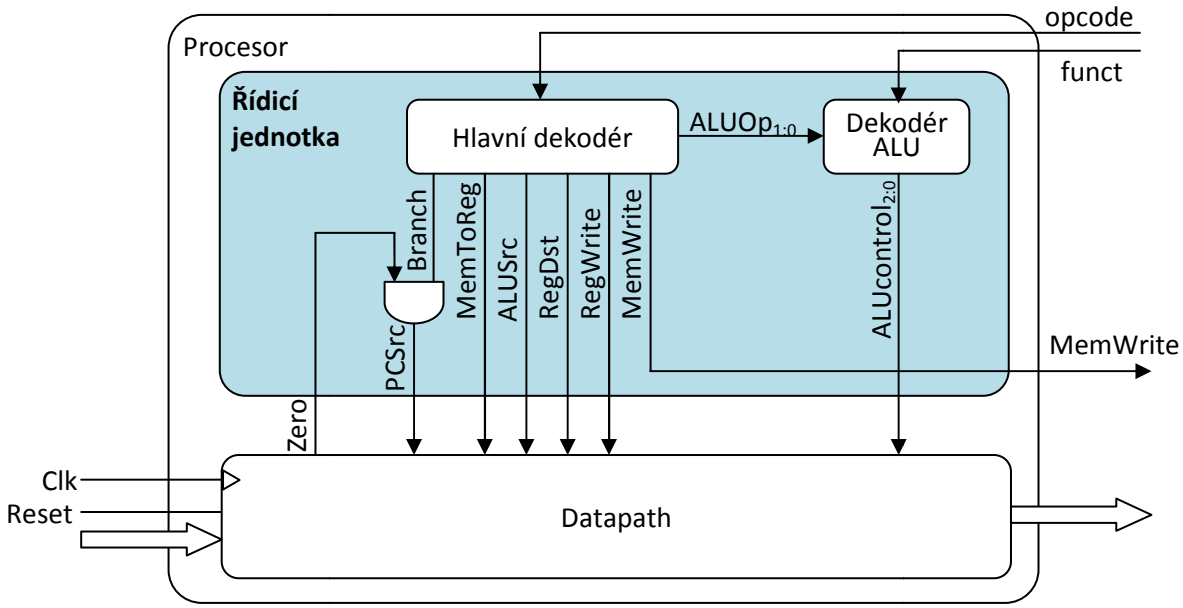
1. Řídicí jednotka zabezpečuje čtení instrukcí z paměti v správném pořadí a jejich dekódování, přijímá stavová hlášení a zabezpečuje správnou interpretaci (vykonání) instrukcí nastavením potřebných řídicích signálů pro funkční jednotky zúčastněné na vykonávání instrukcí. Činnost řídicí jednotky se s výhodou popisuje konečným automatem, zejména při multicyklovém vykonávání instrukcí. Jednotlivým stavům automatu přináleží konkrétní nastavení řídicích signálů (jednotlivé fáze vykonání instrukce při zohlednění typu vykonávané operace), přechodům pak podmínky (stavová hlášení, typ instrukce,...), při kterých se mezi těmito stavy přechází. Nicméně, při jednocyklovém vykonávání instrukcí je možné činnost řídicí jednotky vyjádřit pomocí pravdivostních tabulek.

Popište v jazyku Verilog řídicí jednotku vyhovující specifikacím uvedeným níže.



Předpokládejme, že řídicí jednotka přijímá kód vykonávané instrukce – *opcode*, pole *funct*, a pouze jedno stavové hlášení z ALU indikující nulovost výsledku operace ALU – příznak *zero*.

Řídicí jednotka nechť je rozdělena do dvou částí – hlavní dekodér a dekodér ALU. Činnost řídicí jednotky je popsána následujícími tabulkami. Hlavní dekodér a dekodér ALU jsou propojeny pomocí ALUOp.

Tab. 1 Hlavní dekodér

vstup	výstup						
Opcode	RegWrite	RegDst	ALUSrc	ALUOp	Branch	MemWrite	MemToReg
000000	1	1	0	10	0	0	0
100011	1	0	1	00	0	0	1
101011	0	0	1	00	0	1	0
000100	0	0	0	01	1	0	0
001000	1	0	1	00	0	0	0

Tab.2 Dekodér ALU

vstup		výstup
ALUOp	Funct	ALUControl
00	X	010
01	X	110
1X	100000	010
1X	100010	110
1X	100100	000
1X	100101	001
1X	101010	111

Dále nechť platí, že řídicí signál PCSrc (PCSrc je výstupní signál řídicí jednotky) je nastaven, pokud jsou nastaveny signál Branch hlavního dekodéru a zároveň stavový signál Zero.
Schéma řídicí jednotky spolu s vyobrazením interakce s datovou cestou procesoru (datapath) je znázorněna na úvodním obrázku.

2. Předpokládejte, že máte k dispozici instrukce add, sub, and, or, slt, addi, lw, sw, beq. Přepište níže uvedený program v HLL pomocí těchto instrukcí.

```
if(a<=b)
    for(int i=0; i!=c; i++)
        a += i;
else
    a = a-b;
while(1)
    ;
```

Předpokládejte, že proměnné a, b a c jsou typu *int* (4B) a jsou již uloženy v paměti na adresách 0x0010, 0x0014 a 0x0018.

Pozn.: Po úspěšném vykonání programu bude patřičně modifikována proměnná a na adrese 0x0010 a program uvázne v nekonečné smyčce.

Add	add $d,$s,$t	$d = $s + $t	Aritmetický součet
Subtract	sub $d,$s,$t	$d = $s - $t	Aritmetický rozdíl
Add immediate	addi $t,$s,C	$t = $s + C (signed)	Pro přičtení znaménkově rozšířené konstanty
And	and $d,$s,$t	$d = $s & $t	Logický součin
Or	or $d,$s,$t	$d = $s | $t	Logický součet
Set on less than	slt $d,$s,$t	$d = ($s < $t)	Testuje, zda první registr je menší než druhý; $d nabývá hodnotu 0 nebo 1
Load word	lw $t,C($s)	$t = Memory[$s + C]	Čte slovo (4 B) z paměti. MEM[$s+C] a následující 3 Bajty; C je offset
Store word	sw $t,C($s)	Memory[$s + C] = $t	Ukládá slovo (4 B) do paměti. MEM[$s+C] a následující 3 Bajty
Branch on equal	beq $s,$t,C	if ($s == $t) go to PC+4+4*C	Jde na instrukci na specifikované adrese, pokud registry jsou si rovny

The register letters *d*, *t*, and *s* are placeholders for (register) numbers or register names. *C* denotes a constant (*immediate*).