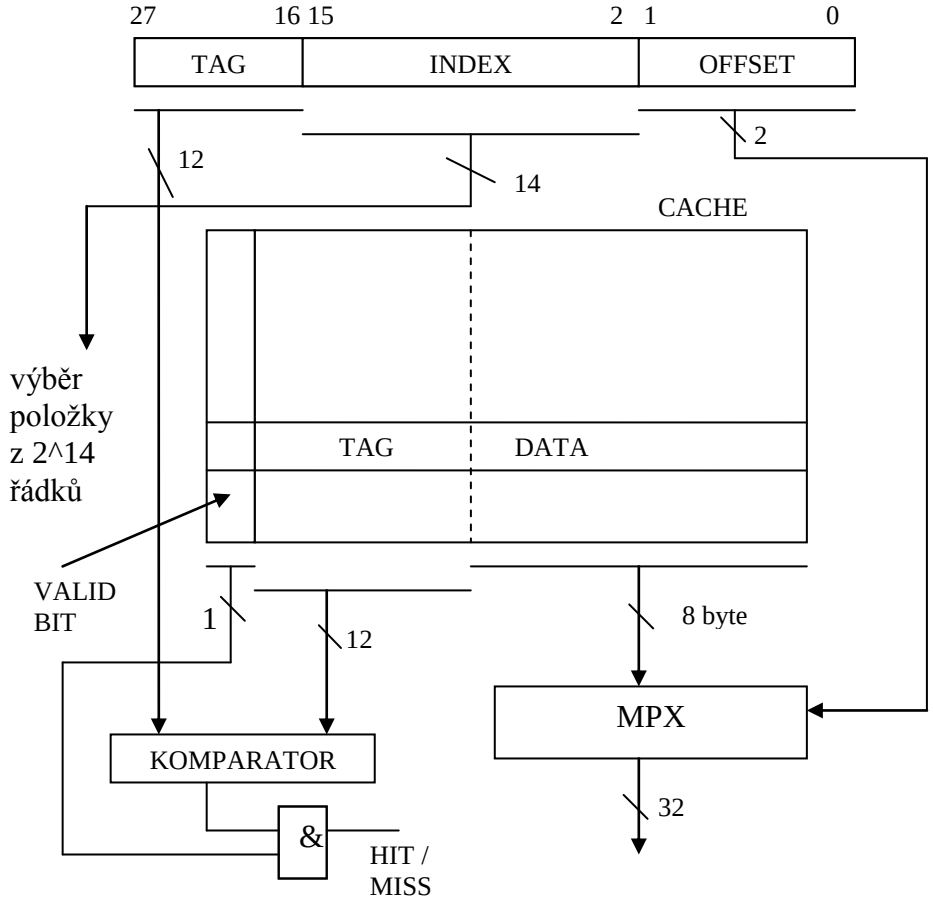
UPA – zkouškové otázky z minulých let

1. 32 bitová sběrnice. Mas pamet o vel. 256MB, cache 128kB s 8byte bloky. Cache je přímá. Mas realizovat cteni. Nakreslit obr, popsat, dimenzovat dat. toky.



Popis: V horní části obr. je adresa, kterou vystavil procesor. Veprostřed je CACHE a vespod nalevo komparátor pro porovnání, zda vybíráme správná data. Napravo dole je multiplexor, který vystaví pouze data podle OFFSET. HIT – udává, že se data v CACHE nacházejí MISS, že ne.

2. Co je hazard (u procesoru typu risc), jak tomu zabranit pripadne to omezit

Hazardy brání provedení následující instrukce v příslušném taktu.

STRUKTURNÍ – různé instrukce se snaží používat současně stejné funkční jednotky

řešení – duplikovat jednotky

ŘÍDÍCÍ – cílová adresa je známa až na konci třetího cyklu => pozastavení

řešení – predikce (statická, dynamická)

- opoždění instrukcí větvení

DATOVÉ – instrukce je závislá na výsledku předchozí instrukce, která je stále v pipeline

- pozastavení – vložení bublin do pipeline

řešení – forwarding = zasílání dat také do dalšího stupně

- změna pořadí instrukcí, aby se omezilo pozastavování

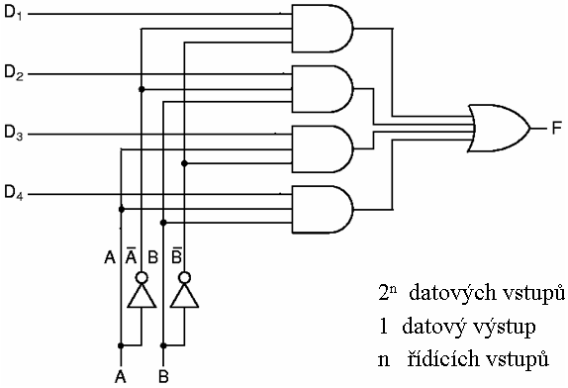
3. V pseudokodu mas pro ruzne rypy architektur realizovat nasledujici operace: a = b + c, b = a + c, d = a – d pro: Stack-machine, Accumulator machine, Load-Store, Memory-register

Udělám pouze pro první příklad:

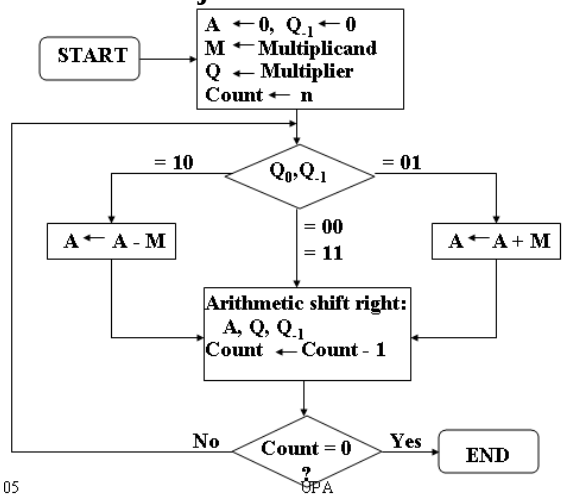
Stack	Accumulator	Load-Store	Memory-register
Push B	Load B	Load R1, B	Load R1, B
Push C	Add C	Load R2, C	Add R1, C
Add	Store A	Add R3,R1,R2	Store A,R1

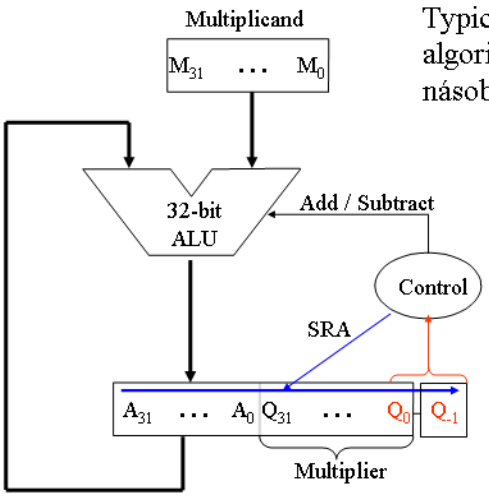
Pop A		Store A,R3	

4. Multiplexor 1 ze 4 - kolik vstupu, vystupu, navrhnout



5. Nasobeni Boothovym algoritmem, princip, jednoduchy vyvojak + zhruba navrhnout hardwarovou jednotku.





Typickým znakem Boothova algoritmu je test dvou bitů násobitele a postup po jednom bitu

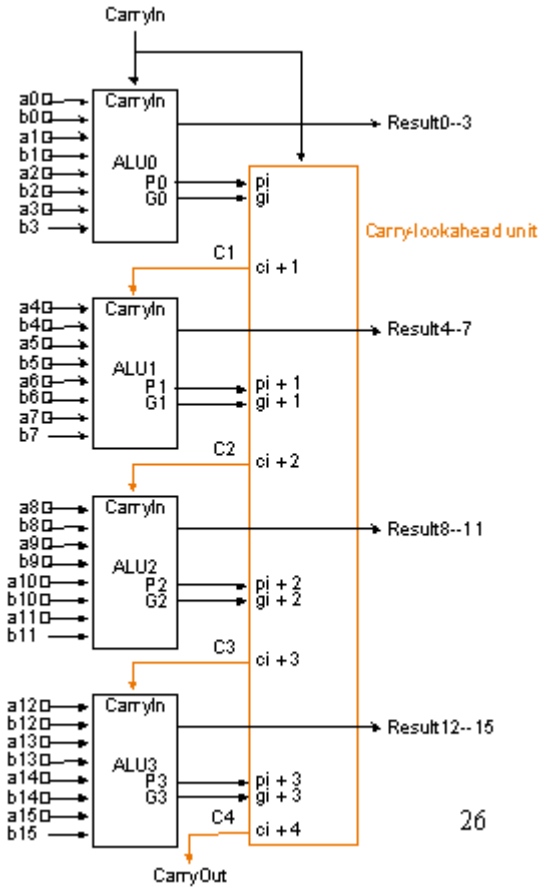
6. Popište funkce Linkeru a jaké datové struktury využívá. [4b]

Sestavuje objektové soubory (.o) a vytváří spustitelný soubor – program. Umožňuje oddělenou kompilaci. Edituje „odkazy“ ve skokových instrukcích, vyhodnocuje reference do paměti. Postup: 1) slučuje kódové segmenty všech .o souborů 2) slučuje datové segmenty všech .o souborů a připojuje je na konec kódových segmentů 3) vyhodnocuje reference. Prochází relokační tabulku a ošetří každou položku (doplní všude absolutní cesty)

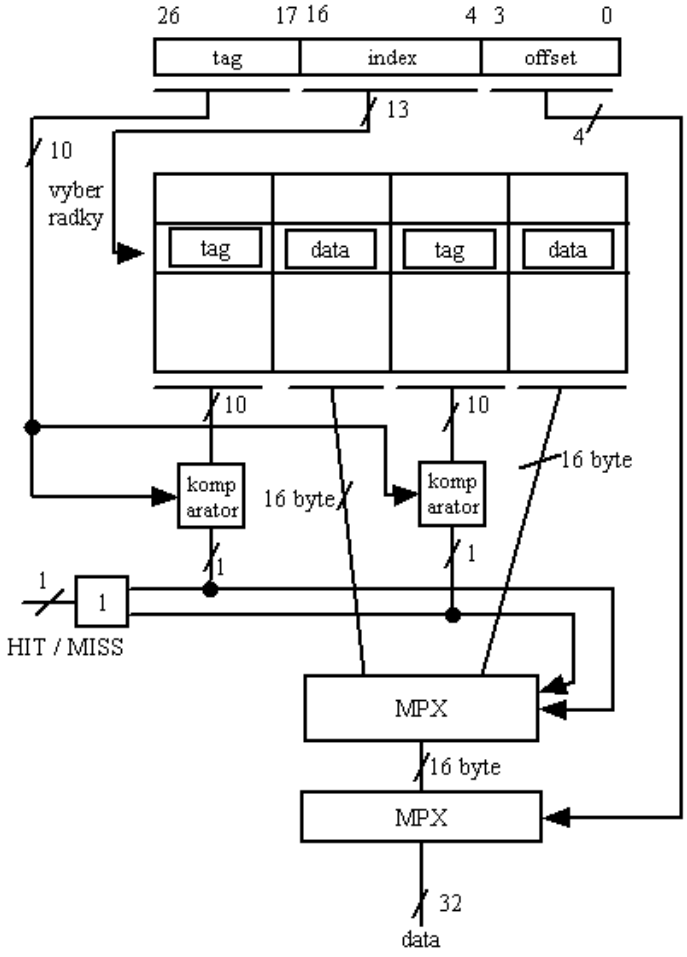
7. Jednodussi procesor, mas tam urcit, k cemu slouzi vyznacena cesta. Mělo se popsat k čemu se asi používá. [4b]

8. Vysvetlete princip urchychleni prace paralelni binarni scitacky. Nakreslete odpovidajici schema pro sirku n-bitu.

Při použití n-bitové sčítačky vytvořené pouze z obvodů 1bitové úplné sčítačky dochází k nepříjemnému jevu. Celým obvodem se šíří signál přenosu. Tato záležitost zpomaluje činnost celé sčítačky. Pro lepší návrh můžeme vyjít ze skutečnosti, že při určitých kombinacích vstupů můžeme rovnou říct, jestli se bude generovat přenos nebo se bude předávat dál. Avšak takové zařízení by bylo velice složité a drahé. Lepší je použít opakovaně několik CLA sčítaček.



9. Pocitac ma hlavni pamet o kapacite 128MB. Cache pamet ma kapacitu 128kB a je organizovana jako dvoucestna "castecne" asociativni cache s velikosti bloku 16Byte. Nakreslete strucne vyberovy mechanismus pri operaci cteni (tim je myslen mechanismus, ketrym se rozpozna, zda pozadovana data jsou v cache pameti a pristup k nim). Dimenzujte spravne jednotlive datove a vyberove linky.



10. Uvedte zakladni charakteristiky instrukcnich souboru procesoru typu RISC. Jakym zpuseobem je organizovan pristup do operacni pameti.

Instrukční soubor lze chápat jako interface mezi SW a HW – představuje abstraktní formu HW. Odděluje celou složitost implementačních detailů od SW. Šest základních typů instrukcí: Load/Store, výpočetní, skoky a větvení, pohyblivá řádová čárka, Memory Management, speciální. Tři formáty instrukcí, každá o délce 32 bitů. Data jsou v paměti zarovnána a přístupná pouze pomocí instrukcí load a store.

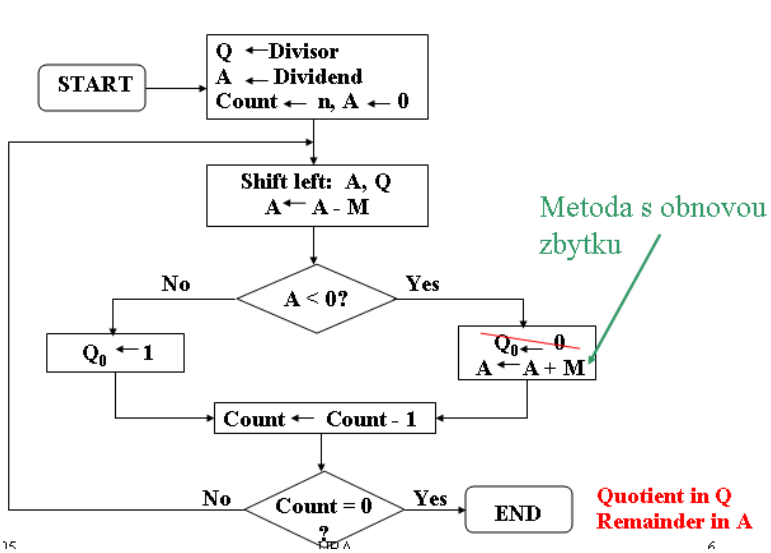
11. Popiste jakym zpusobem se zobrazuji cisla v pohyblive radove carce. Jednotilive slozky cisla jsou ulozena ve slove v urcitem poradi, uvedte duvody.

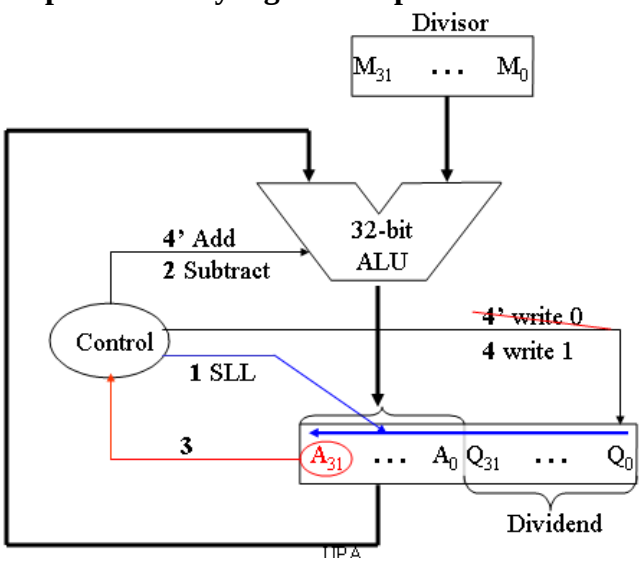
- Normalizovaný formát: $+1.xxxxxxxxxx_2 * 2^{yyyy_2}$
- Násobek délky slova (32 bitů)

- S je znaménko
  Exponent je znázorněn yyyy
  Mantisa je znázorněna xxxx

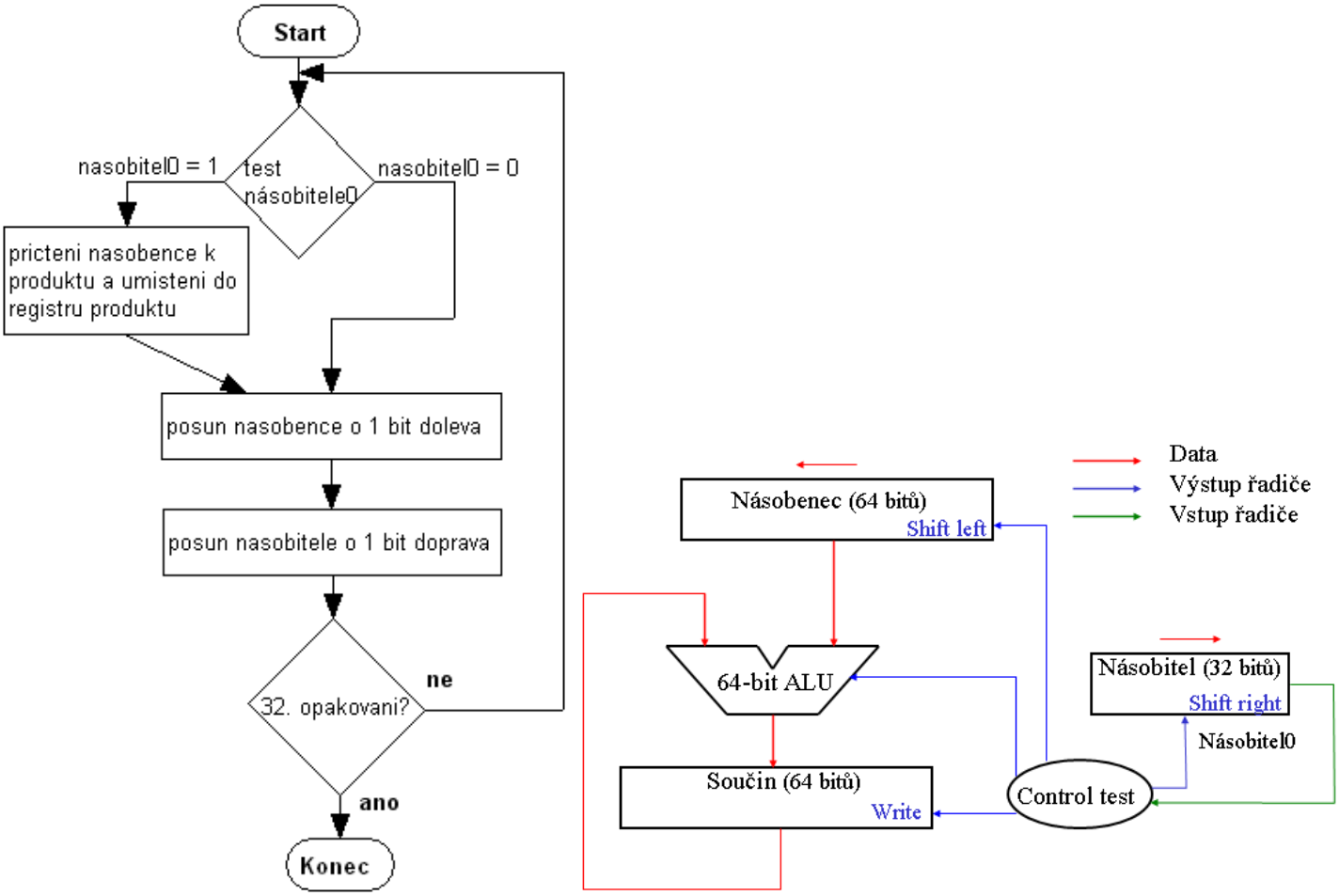
Důvodem je rychlé porovnávání, když nemáme k dispozici FP jednotku. Pak se porovnává znaménko, následně exponent a na závěr mantisa => někdy šetří čas.

12. Popiste algoritmus pro deleni binarnich cisel bez znaménka, nejlepe formou vyvojoveho diagramu. Nakreslete operacni jednotku, ktera by byla schopna navrzeny algoritmus provadet.





13. Formou vyvojoveho diagramu vysvetlete klasicky algoritmus operace nasobeni binarnich cisel s testem jednoho bitu a s posuvem rovnez o jeden bit [2] navrhnete operacni jednotku pro provadeni tohoto alg. pokuste se zduvodnit z ktere strany se obvykle postupuje pri testovani nasobitele [2]



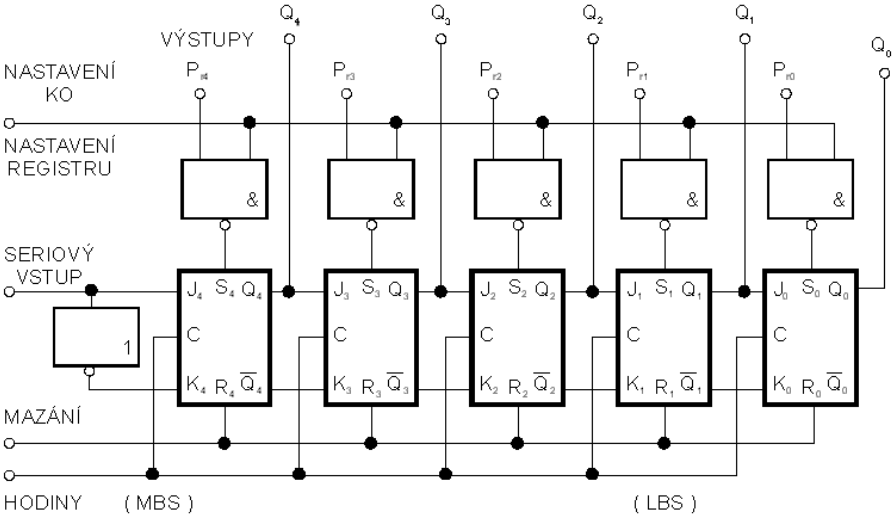
Při násobení dvou 32 bitových čísel vzniká 64 bitový výsledek. Naše varianta řešení využívá pro výpočet výsledků 3 64 bitové registry. Abychom ušetřili, lze použít 3 32 bitové registry. Násobenec a násobitel jsou uloženy do 32 bitových registrů. Zároveň tvoří poslední volný registr a registr, kde je uložen násobitel celkový výsledek. Pak se vždy testuje 0. bit násobitele a výsledek se přičítá do 3. registru. Po každém cyklu se posune registr násobitele a 3. registr o jeden bit doprava vypadnutý bit z 3. registru se doplní do registru násobitele na nevyšší bitovou pozici. Celkový výsledek je pak uložen ve dvou 32 bitových registrech.

14. Vysvetlete a pojmenujte na obrazku uvedeny adresovy rezim. Pro ktere objekty a konstrukce je tento adresovy rezim vhodny? [4]

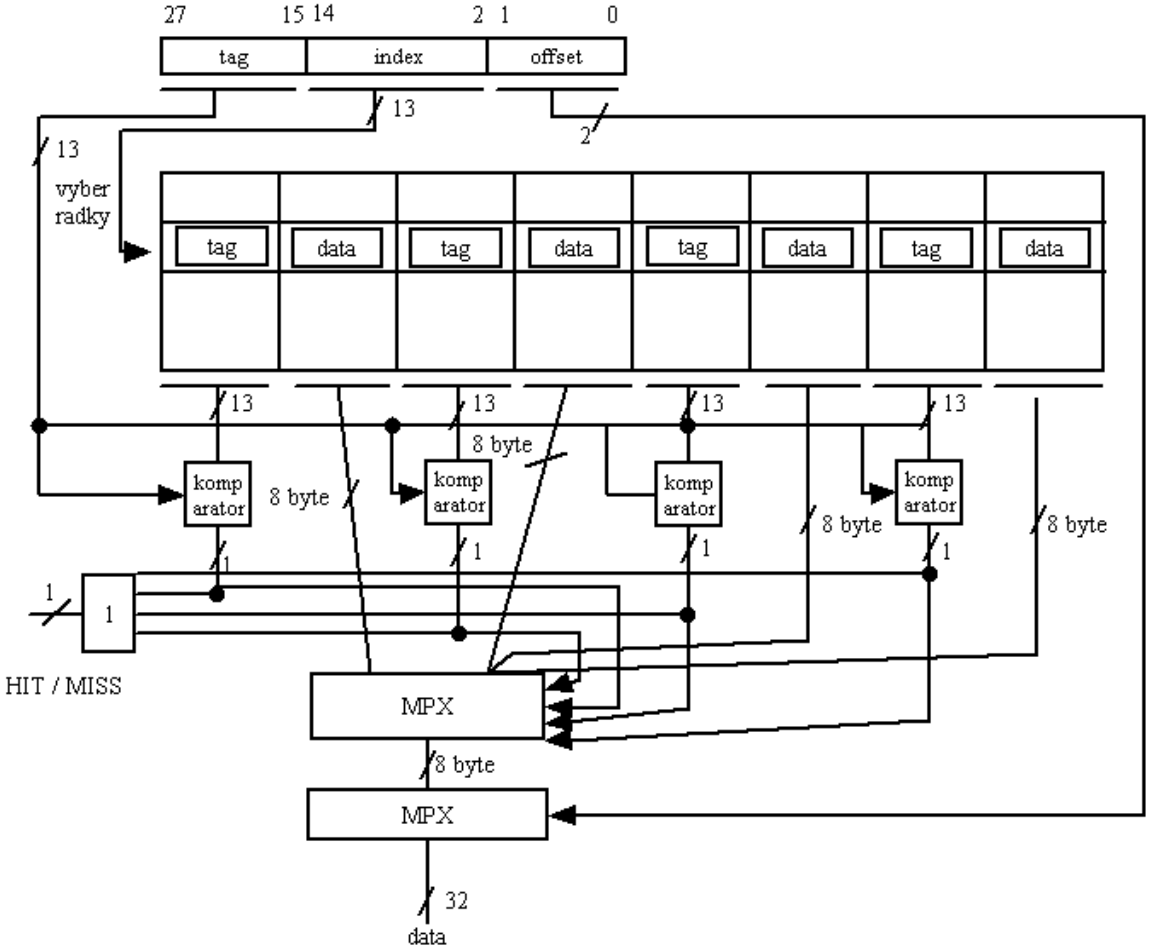
15. vysvetlete princip mikroprogramoveho rizeni (mikroprogramovy automat a jeho strukturu). [4]

Myšlenka je v řízení činností procesoru pomocí mikroprogramu. Mikroprogram je uložen v ROM nebo PLA. Mikroprogram dovoluje skoky a větvení. Vybírá se mikroinstrukce za mikroinstrukcí a podle ní se provádí činnost procesoru. Usnadňuje změnu řízení procesoru. Není rychlejší. Usnadňuje návrh řízení (mnohdy je těžké až nemožné vytvořit konečný automat, podle kterého by se činnost procesoru řídila).

16. navrhnete fcni blok "posuvny registr" o delce 5 bitu. Pro jeho navrh vyberte libovolny typ klopneho obvodu. [4]



17. hlavni pamet == 256MB, cache == 64KB je organizovana jako ctyrcestna "castecne" asociativni cache s velikosti bloku 8 byte. nakreslete a popiste strucne vyberovy mechanizmus pri operaci cteni (tim je myslen mechanizmus, kterym se rozpozna zda pozadovana data jsou v cache pameti a pristup k nim). dimenzujte spravne jednotlive datove a vyberove linky. [4]



18. vysvetlete vyznam a fci bloku, ktery je na obrazku vyznacen. jakym zpusobem je pouzivan. [4] (obr == http://www.kiv.zcu.cz/~vavricka/UPA/Opravene_obr/F0630.pdf a oznacen byl ctverec ve fazi decode (druhy ctverec zleva))

19. uvedte co je to prerusovaci vektor, o jaky typ informace se jedna a jak je v systemu pouzivan. kde ho lze vlastne nalezt? [4]

Na začátku paměti je tabulka s adresami obslužných program. Položky tabulky se nazývají vektory přerušení. Při každém požadavku na přerušení se uloží stav procesoru a přejde na adresu obslužného programu. Po dokončení tohoto programu se předchozí činnost obnoví a výpočet pokračuje dál.

20. vysvetlete co je to system dynamicke transformace adresy (virtualni pamet). [2] uvedte alespon jeden typ transformacniho mechanizmu. [2]

Procesor pracuje s tzv. virtuálními adresami. Tato adresa se musí transformovat na fyzickou adresu do paměti. To provádí Memory management unit. Systémy virtuálních pamětí používají několik technik.

Mechanizmus stránkování – virtální adresní prostor se rozdělí na stránky pevné velikosti. Fyzický prostor se rozdělí na rámce stejné velikosti. Stránka může obsahovat pouze jeden rámec. Na známém místě je uložena mapa stránek. Tato tabulka slouží k mapování virtuálních stránek na rámce.

21. Popište způsoby zápisu dat do paměti, pokud je v systému přítomna cache.

1) Write-Through:

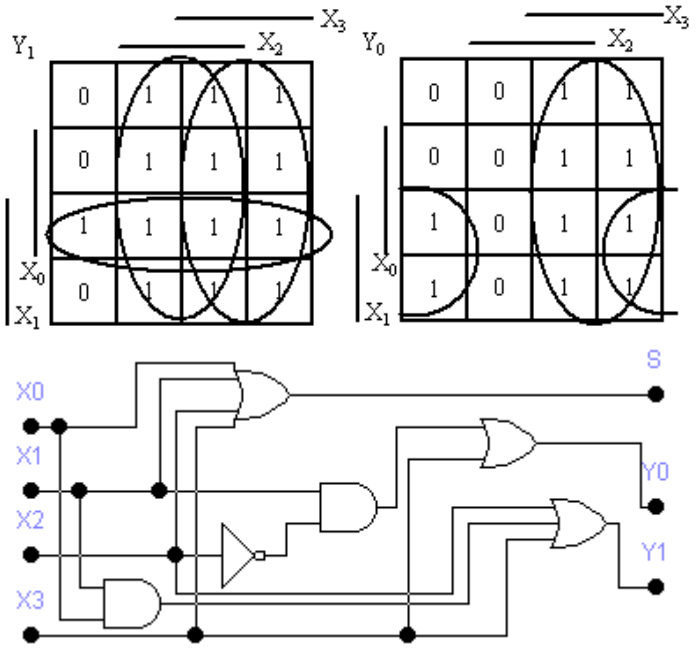
- Zápis dat (a) do cache *a současně* (b) do bloku v hlavní paměti
- **Výhoda:** Případ „Miss“ je jednodušší & levnější, protože není třeba zapisovat blok zpět do nižší úrovně
- **Výhoda:** Snazší implementace, je třeba pouze zápisový buffer

2) Write-Back:

- Zápis dat *pouze* do bloku cache. Zápis do paměti pouze při výměně bloků
- **Výhoda :** Zápisy omezeny pouze rychlostí zápisu cache
- **Výhoda :** Podporovány zápisy více slov najednou, efektivní je pouze zápis celého bloku do hlavní paměti najednou

22. Navrhněte kombinační logický obvod "priorotní funkce". Obvod má 4 vstupy (číslované od 0 do 3) a 3 výstupy. První výstup je v 1, pokud byla alespoň na jeden vstup přivedena 1. Na dalších dvou výstupech se objeví číslo vstupu na němž je přivedena 1. Pokud je 1 na více vstupech, pak na výstupu číslo vstupu s nejvyšší prioritou.

X_3	X_2	X_1	X_0	Y_1	Y_0
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	1
0	0	1	1	0	1
0	1	0	0	1	0
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	1	1	0
1	0	0	0	1	1
1	0	0	1	1	1
1	0	1	0	1	1
1	0	1	1	1	1
1	1	0	0	1	1
1	1	0	1	1	1
1	1	1	0	1	1
1	1	1	1	1	1



23. Předpokládejme mikroprocesor s cache, který pracuje následujícím způsobem: pokud je třeba zavést blok, pak je zaveden, provede se výpočet a pak se vyhodí některý ze starých bloků. Rozeberte možnosti použití jednotlivých probíraných implemetací cache pro takový mikroprocesor.

Přímo mapovaná cache – lze těžko implementovat. Nemáme informaci o tom, který blok je starý. Museli bychom vybírat náhodně.
Čistě asociativní cache – velice vhodná. Máme informaci o tom, které bloky jsou jak staré. Nevýhodou je cena.
vícecestná částečně asociativní cache – Stejný problém jako přímomapovaná

24. Jakým způsobem je volán podprogram v mikroprocesoru. Jaké operace je třeba provést.

Těsně před vyvoláním funkce volající
- Předá argumenty ($a0 – $a3). Další arg.: uloží do stacku
- Uloží ukládané registry volajícího ($a0 – $a3; $t0 – $t9)
- Provede instrukci jal (skok na volanou proceduru a uložení návratové adresy)

Těsně před zahájením výpočtu volané funkce se
- Alokuje paměť pro frame ($sp = $sp - fsize)
- Uloží ukládané registry volaného ($s0-$s7; $fp; $ra)
- $fp = $sp + (fsize-4)

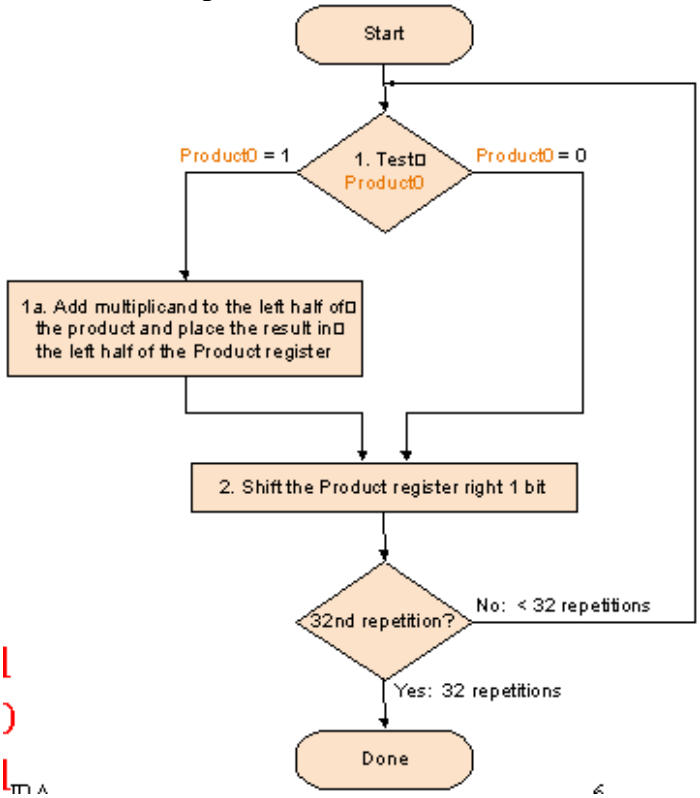
Těsně před návratem do volajícího:
- Uložení funkční hodnoty do registru $v0
- Obnovení všech registrů volající funkce
- Pop stack frame ($sp = $sp + fsize); obnova $fp

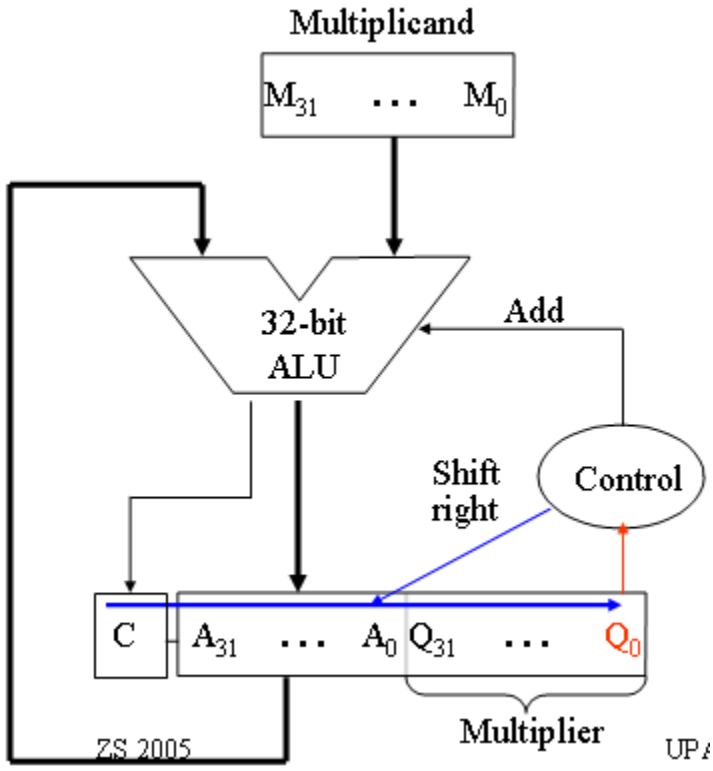
- Návrat provedením skoku na adresu uloženou v $ra

25. Vyjmenujte a popište situace, za jakých se může změnit registr PC (Program Counter).
PC se zvyšuje při načtení instrukce.
Při skoku a větvení.
Při volání podprogramu.
Při návratu z podprogramu.

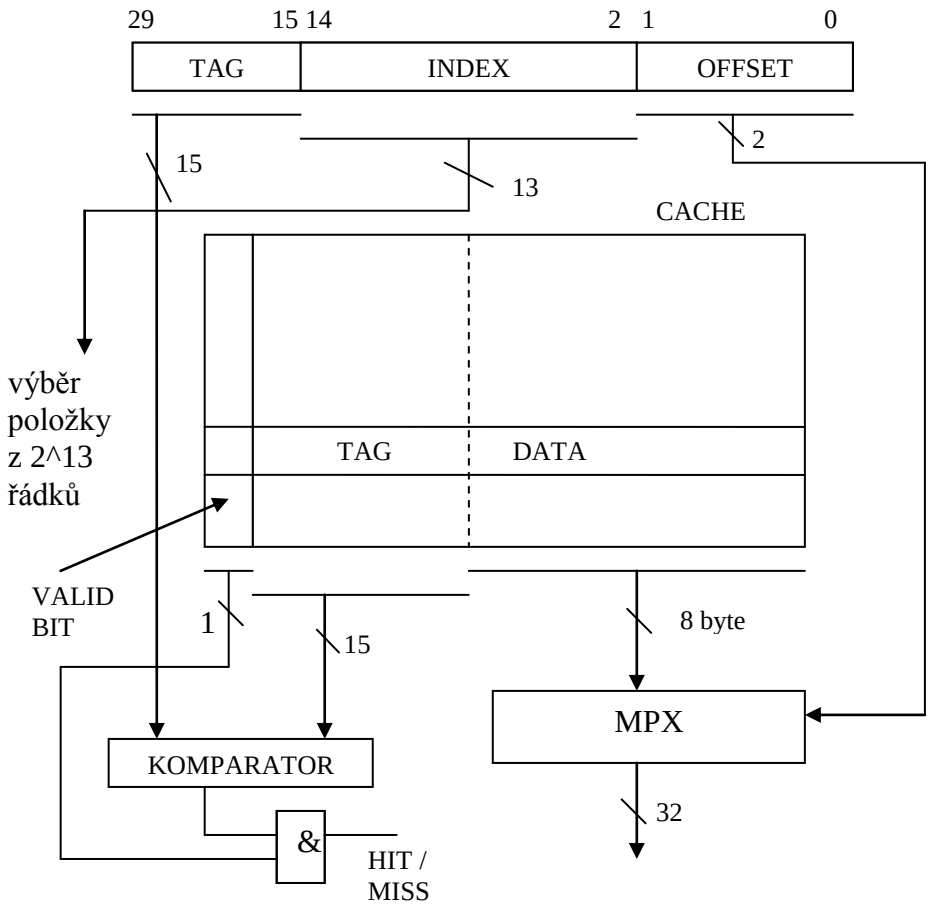
26. Jakým způsobem se sčítají 2 desetinná čísla v počítači.
Nejdříve se musejí převést na stejný exponent. Poté se sečtou mantisy a vynormují zpět.

27. Nakreslete vývojový diagram pro násobení dvou kladných čísel v počítači. Nakreslete obvod pro realizaci této operace.





28. PC má hlavní paměť 1GB a Cache má mít velikost 64KB a je organizována jako přímo mapovaná cache s velikostí bloku 8byte. Nakreslete a stručně popište výběrový mechanismus při operaci čtení. [4b]



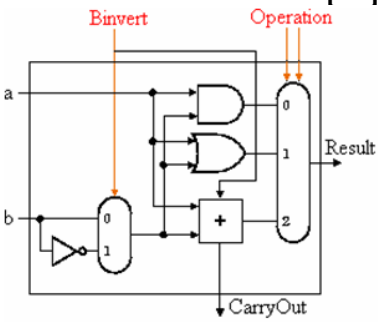
29. Navrhněte logický obvod s použitím hradel, který má 3vstupy a 7 výstupů a je charakterizován funkcí: "Počet aktivních výstupů je roven binárnímu číslu na vstupu". Nakreslete schéma. [2b návrh 2b schéma]

viz cvičení. Nebo jde použít dekodér 1 z N (pomocí hradel) a ten upravit.

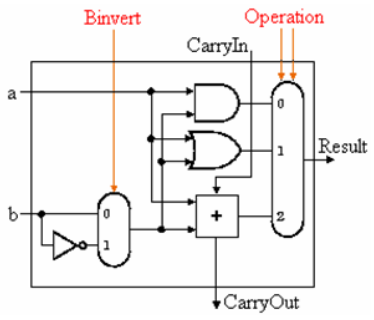
30. Popsat zadane operace pomoci ctyr ruznych architektur.

a=(b+c)*c
b=a*c+c
d=(a-d)/(a+d)
pro Stack-machine, Accumulator machine, Load-Store, Memory-memory [vše po 1bodu]

31. Doplňte paralelní binární sčítačku čísel v doplňkovým kódu pro operace (+,-). +,- se volí vstupním signálem. Zdůvodněte. PS: Zde bylo napsáno doplnit, ale nebylo tam do čeho, prostě to musíte celý navrhnout a načmárat. [4b]

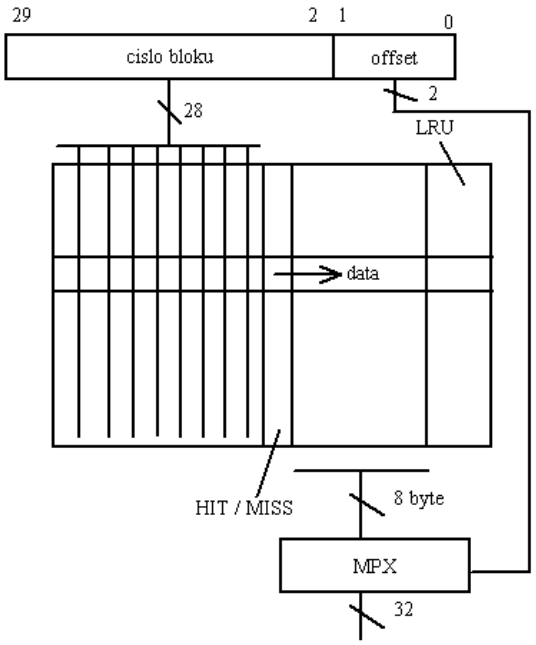


Bit s nejnižší váhou



Ostatní bity

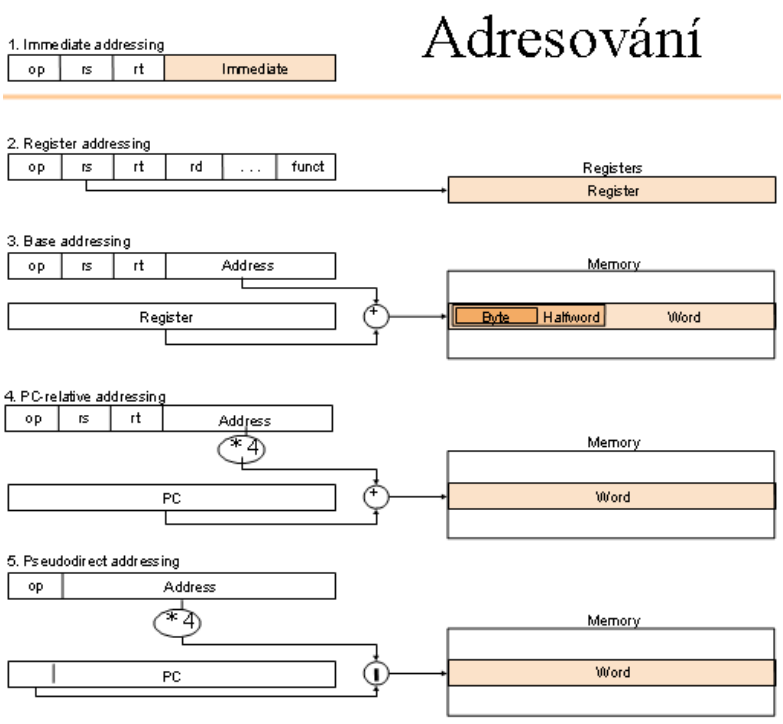
32. Pamet 1GB, cache 64kB s 8B bloky. Vhodne dimenzovat datove toky, vyberovy mechanismus pri cteni a zapisovani. Cache byla myslim plne asociativni.



33. Co je to ortogonalita operacniho kodu, adresy a jeste neco, to si bohuzel nepamatuju :(

Ortogonalita operačního kódu znamená, že všechny instrukce, které pracují s daty, mají možnost pracovat se všemi typy dat (long, word, byte).

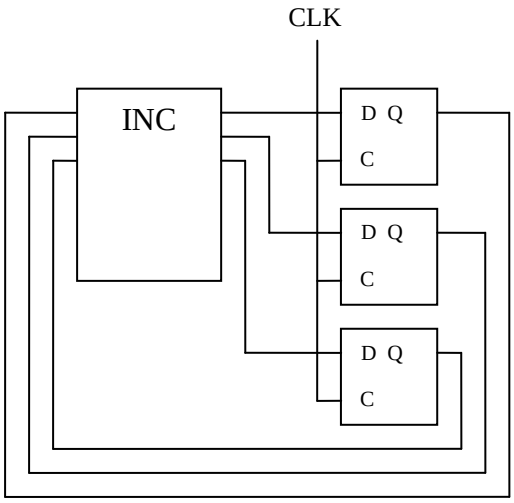
34. Nejakej obrazek s PC a instrukci a odkaz nekam do pameti, napsat co je to za adresni rezim a k cemu se da vyuzit (nebo tak neco).



- Registrové adresování
  - Nejobvyklejší způsob (nejrychlejší a nejkratší)
  - add $3, $2, $1
- Bázované adresování
  - Operand je v paměti na místě udaném offsetem
  - lw $t0, 20 ($t1)
- „Immediate“ operandy
  - Operand je malá konstanta uvnitř instrukce
  - addi $t0, $t1, 4 (16-bit integer se znaménkem)

35. Navrhnout synchronni 3bitovej citac.

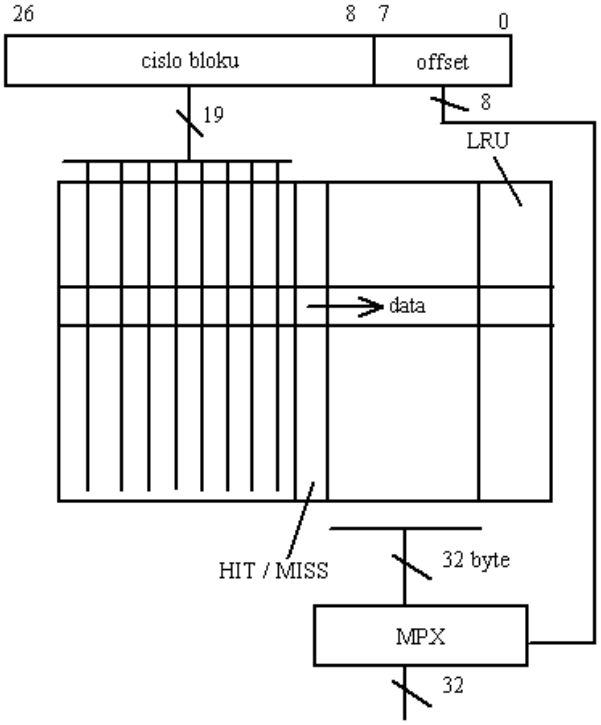
Použijeme inkrementační obvod a 3 KO typu D.



36. Co je to radic procesoru - popsat a vysvetlit a o jakej logickej obvod se jedna.

Řadič procesoru je jednotka, která se stará o řízení činností procesoru. Je to konečný automat. Je implementován jako programovatelné logické pole – obsahuje přechodovou funkci a registr, ve kterém je uložen stav konečného automatu.

37. plne asociativni CACHE, 128MB, 128KB, 32K



38. Periferní přenosy / HW

39. Rozdíly mezi jedno, dvou a částečne asociativní čtyřcestnou cachí.

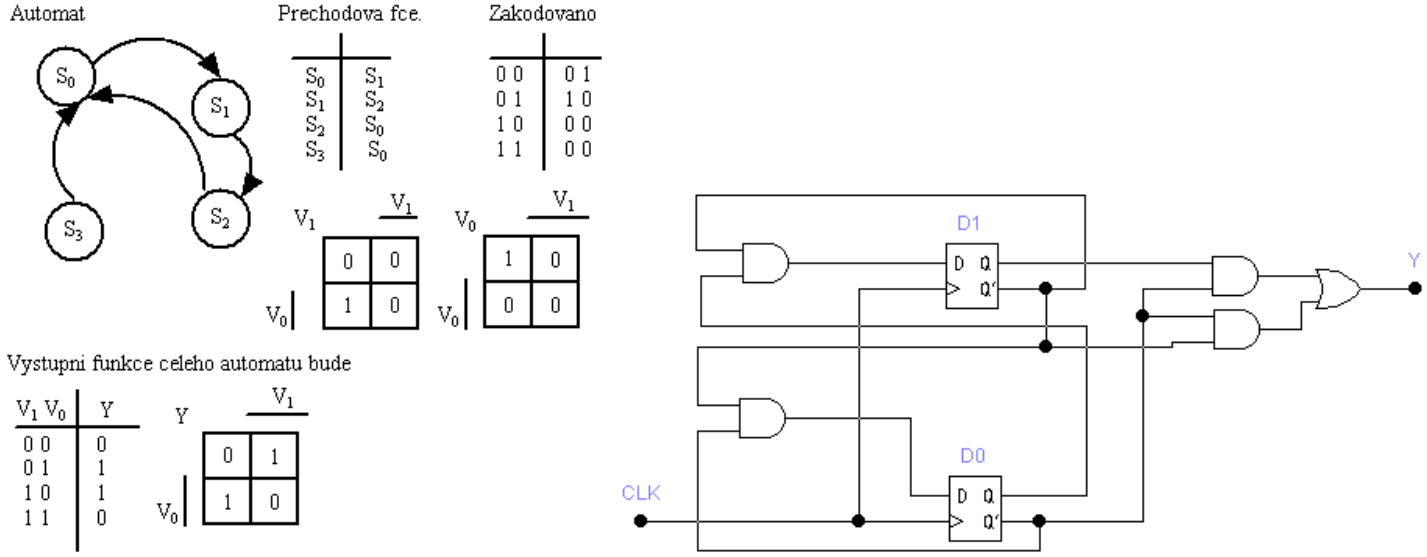
Částečně asociativní mapování

- **Zobecňuje všechna mapovací schémata cache**
  - Předpokládejme, že cache obsahuje *N* bloků
  - 1- cestná částečně asociativní cache: Přímé mapování
  - *M*-cestná částečně asociativní cache: Je-li M = N, pak se jedná o plně asociativní cache
- **Výhoda**
  - Zvyšuje úspěšnost – klesá „miss rate“ (více míst, kde lze nalézt B)
- **Nevýhoda**
  - Narůstá „hit time“ (více míst, kde je třeba hledat B)
  - Složitější hardware

40. Na vstupu pouze časovač. Navrhněte co nejjednodušší obvod z klopných obvodů s výstupem: 011011011...

Jde o to, že si vytvoříme automat, který má 3 stavy a pak z těch stavů uděláme výstup, který bude 0 nebo 1 a bude se měnit s hodinami. Jako klopný obvody si zvolím např. D.

Přechodová funkce je:



41. Co se děje při volání instrukce.

42. Pointery v C *p=185 - napište kód pro Assembler, typ datové výměny.

`c` je `int`, má hodnotu 100, v paměti na adrese 0x10000000,
`p` je v `$a0`, `x` je v `$s0`

```
1. p = &c;    /* p gets 0x10000000*/
    lui $a0,0x1000 # p = 0x10000000
2.  x = *p;  /* x gets 100 */
    lw  $s0, 0($a0) # dereferencing p
3. *p = 200; /* c gets 200 */
    addi $t0,$0,200
    sw   $t0, 0($a0) # dereferencing p
```

Tady jsou uvedeny všechny případy práce s pointerama. Pro naše zadání by stačilo nejspíš 1 a 3.

43. Výhody a nevýhody procesorů RISC.

Výhody:

- RISC procesory jsou rychlejší (při srovnatelném kmitočtu)
- Jednodušší hardware
- Rychlejší návrh procesoru
- Nižší náklady na vývoj a výrobu

Nevýhody:

- Programy pro RISC jsou delší a komplexnější, než pro CISC
- Programátoři musejí dávat pozor, aby procesor často nečekal dlouho na dokončení instrukce
- Vyžaduje velmi rychlé paměti pro rychlé načítání instrukcí