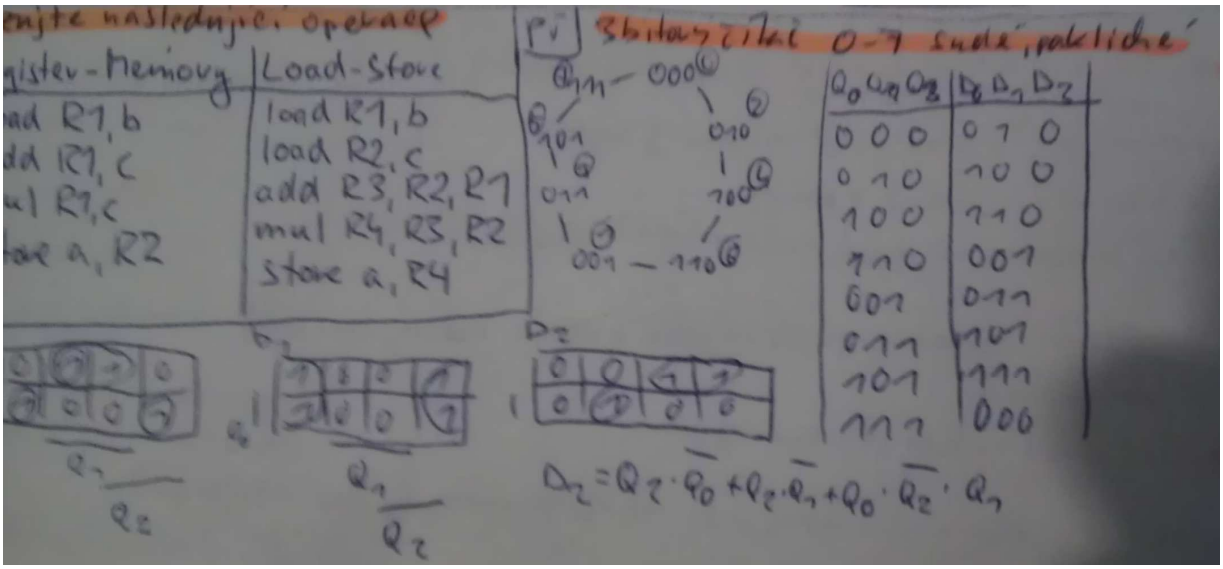
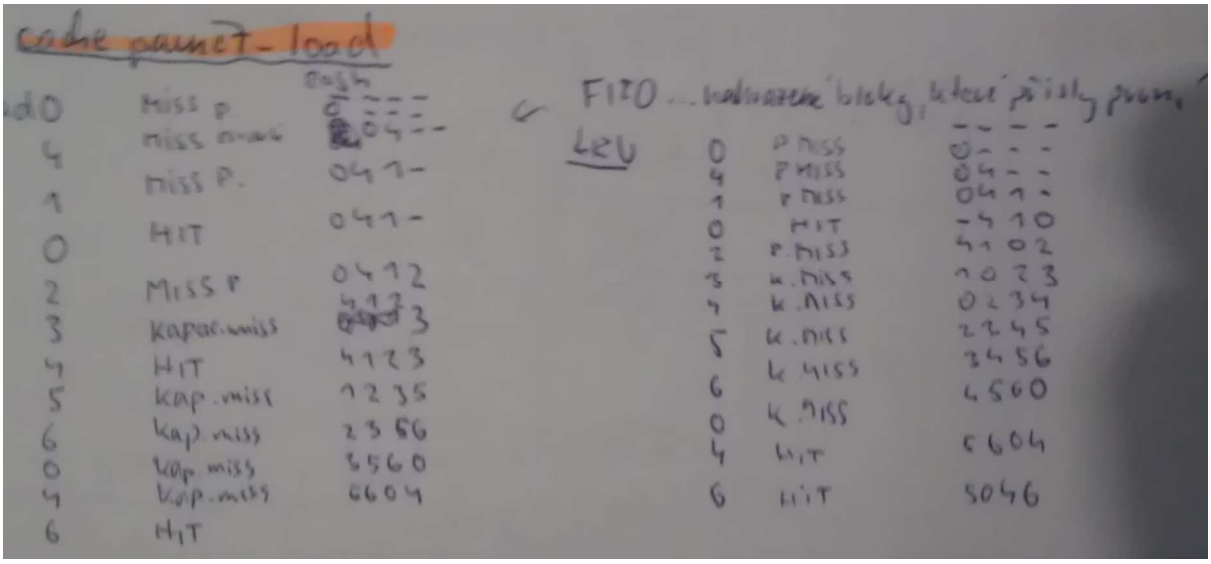
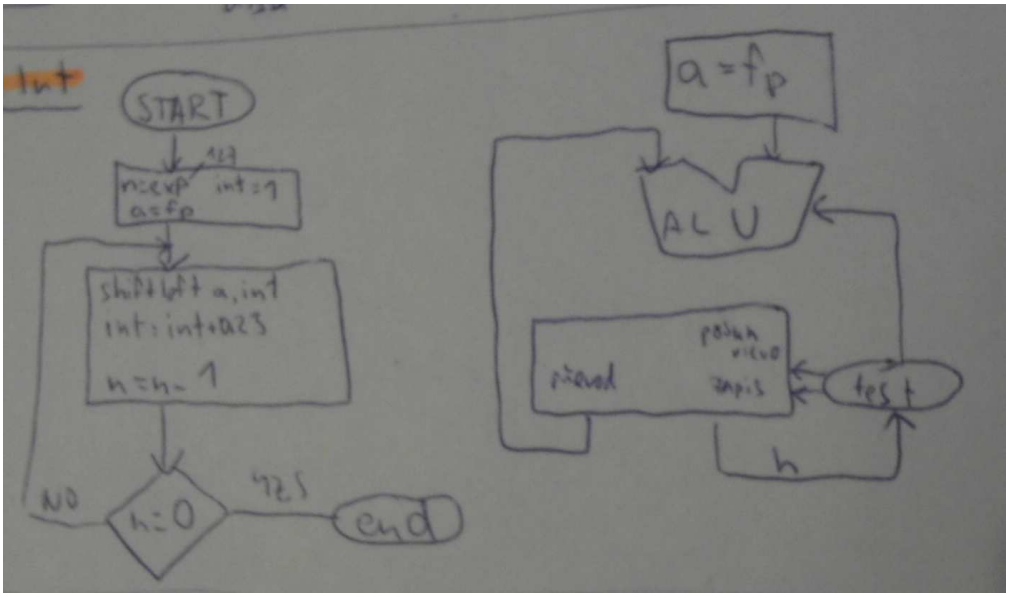
2.Navrhněte jednoduchý sekvenční synchronní automat – tříbitový čítač, který na svém výstupu generuje periodicky čísla od 0 do 7, nejdříve lichá, pak sudá. Paměťový prvek zvolte.



3.Předpokládejte cache paměť o velikosti 4 bloky. Pro následující sled operací „Load“, kde Load X znamená načíst byte z bloku počínajícího na adrese X, označte každý „Load“ jako „HIF“, povinný „MISS“, konfliktní „MISS“ a nebo kapacitní „MISS“. Předpokládejte, že cache je zpočátku prázdná. Pro každý přístup uveďte typ(jeden z výše uvedených) a pak obsag paměti. Otázku řešte pro následující typy cache paměti: (a)Plně asociativní cache se strategií FIFO



5.FLOAT DO INT

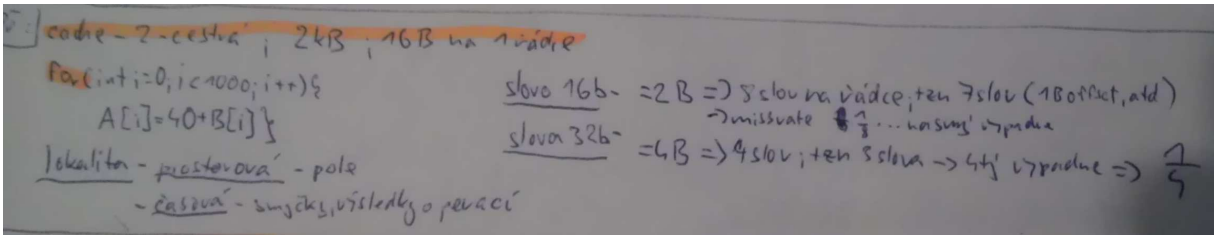


6. Jaké znáte metody dynamické transformace adresy (virtuální paměť). Míněn je přechod od logické adresy (virtuální-tu, kterou generuje procesor na základě běhu programu) na fyzickou adresu (adresa, která je použita k přístupu do fyzické paměti). Cache paměť neuvažujte. Popište (nebo lépe nakreslete) transformační mechanismy.

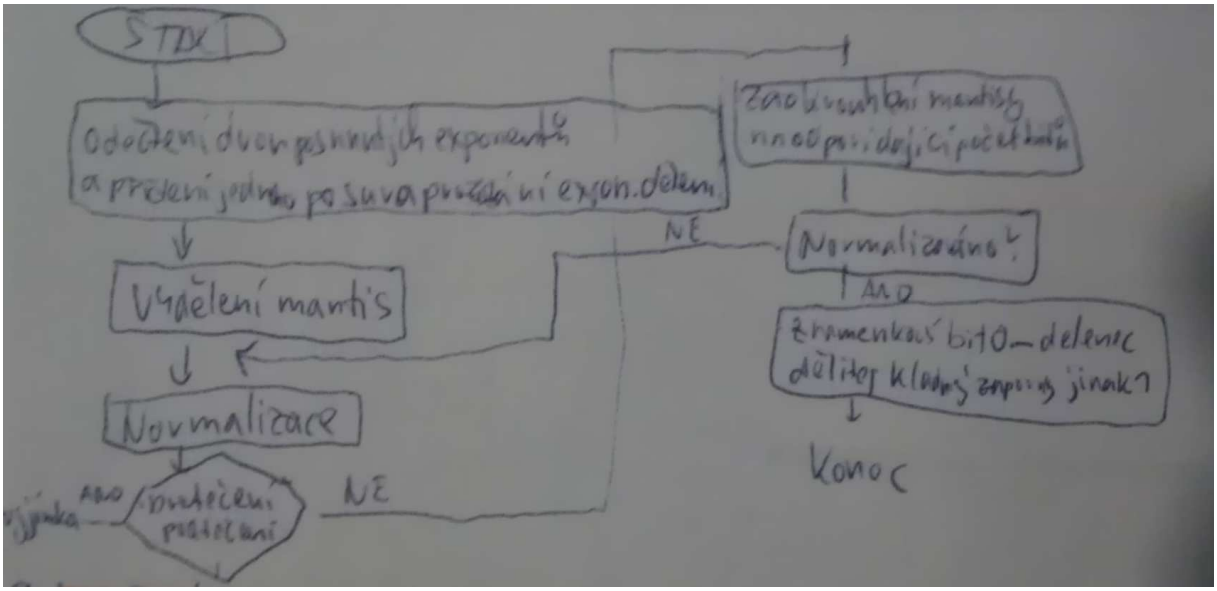
Přímý – stránkování – dva přístupy, z tabulky stránek na disk a pak do paměti. Dvojnásobný čas

S TLB – prvně do tlb a hned do paměti. Když není v tlb tak pak do tabulky stranek

7. Je dána dvoucestná cache paměť o kapacitě 2kbyte se 16 byte na jedné řádce a následující kód: a) Vypočítejte celkovou četnost výpadků („missrate“) (pro případy, že každý prvek pole zabírá jedno slovo 16 bitů, pak také 32 bitů) b) Jaké typ lokality je využíván?



8. Popište, jakým způsobem se dělí čísla v pohyblivé řádové čárce.



9. Vysvětlete, co jsou zarovnaná (nezarovnaná) data, popř. instrukce. Existují procesory, které s nezarovnanými daty popř. instrukcemi dokážou pracovat, jiné typy to nepřipouští. Porovnejte obě varianty (+-), jaké jsou výhody a nevýhody obou řešení.

-adresa v paměti A je N-bajtově zarovnána, když N je mocnina dvou, A je násobkem N bajtů. Bajt je tedy nejmenší jednotka pro přístup do paměti. Každá adresa určuje jiný bajt.

Výhoda: snadno se adresuje. Nevýhoda: mohou zůstat díry tím, že ukládám něco malého a nevyužiji rozsah bajtu

---

1) Jakým způsobem se předávají parametry z hlavního programu do podprogramu z technického hlediska. Jako příklad můžete použít R3000.

- první 4 parametry v registrech $4-7, tj. $a0-$a3, první parametr procedury je vždy v $a0. Zbylé do zásobníku.

2) Co je příčinou fyzických limitů Mooreova zákona? Proč?

-nutná izolační vzdálenost 2-3 nm, omezen kvantovými tunelovými efekty

3) a) Diagram adresního režimu pro výpočet adresy cíle podmíněného skoku.

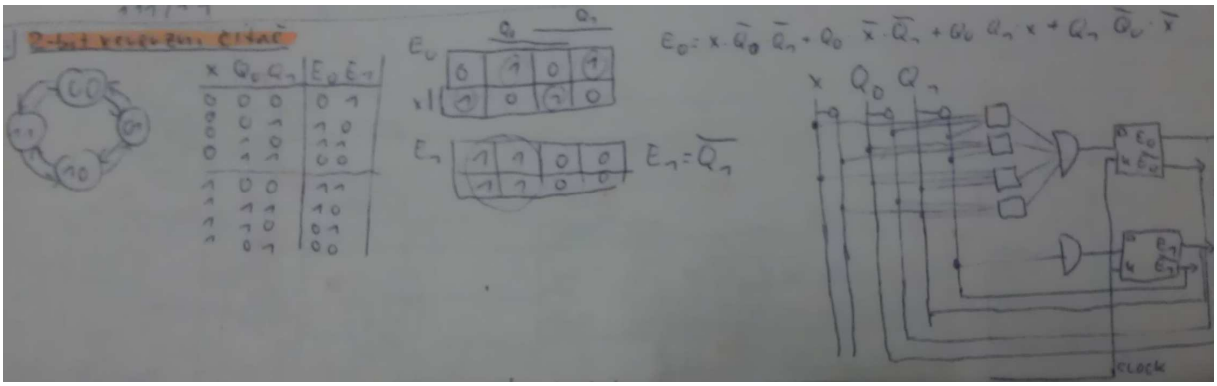
b) Proč jsou instrukce MIPS lw/sw zařazeny do I-formátu instrukcí? Jaké to přináší výhody pro návrh MIPS ISA? protože I je formát pro přesuny dat, větvení, immediate a lw/sw jsou instrukce pro přenos dat

c) Příklad MIPS kódu, který reprezentuje dereferencování pointeru v lw/sw. lw $s0, 0($a0)

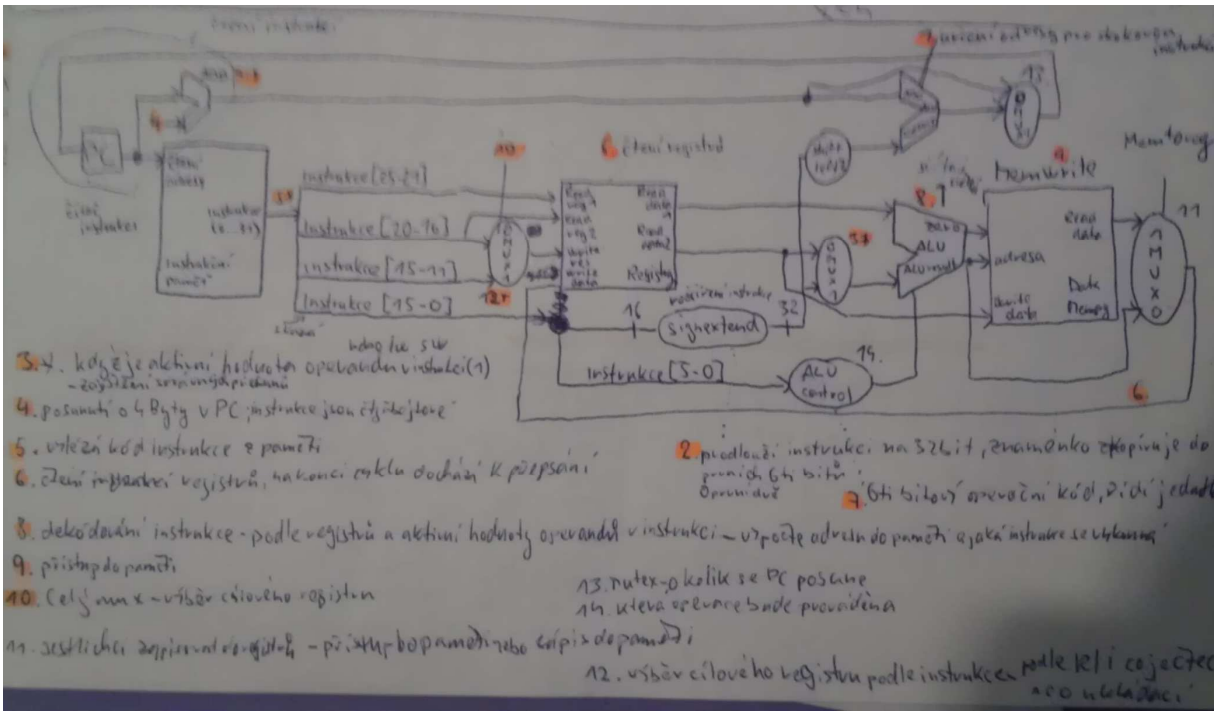
4) Popište podrobně kroky, které procesor vykonává při provádění instrukce návratu z podprogramu. Začněte přesunem instrukce do instr. registru.

- Před návratem do volajícího – uloženi fčni. hodnoty do reg $v0 → obnoveni všech reg. Volané fce.→ pop stack frame + obnova $fp → návrat na adresu uloženou v $ra

5) Navrhněte jednoduchý sekvenční synchron. automat – dvoubitový binární reverzní čítač



6) Obrázek



7) K čemu se využívají příznak modifikace a příznak platnosti příslušného bloku cache paměti. Kdy a z jakých důvodů se nastavují popř. nulují.

-Příznak platnosti přísl. Bloku. Kdy a jak? Když se zapne počítač, nastavují se na 0, při startu procesoru bývá cache naplněna náhodnými daty- i po chvíli činnosti nemusí být obsah všech položek v cache platný-> přidání bitu platnosti (valid bit) ke každému bloku

Příznak modifikace: Indikuje, zda byla data korespondujícího bloku paměti modifikována, či nikoliv. Tento bit je nastaven, když procesor zapisuje do paměti. Obojí se asi nulují při zapnutí počítače

8) Popište algoritmus pro převod čísel ze zobrazení v pohyblivé čárce float (jednoduchá přesnost) do formy zobrazení v pohyblivé řádové čárce double

??? shit

9) Předpokládejme, že čítač M1 má frekvenci hodin CPU 500MHz a 4 třídy instrukcí. Třída A má CP1 = 1, třída B má CP2 = 2, třída C má CP3 = 3, třída D má CP4 = 4. Program obsahuje 20% instrukcí třídy A, 30% B, 10% C. 40% D. a) Jaká je střední hodnota parametru CPI pro program P? b) Bude-li program P mít 10 000 instrukcí, jaký bude celkový počet cyklů spotřebovaných pro běh programu a jaký bude celkový čas pro P na stroji M1?

a) CPI = 1*0,2+2*0,3+3*0,1+4*0,4 = 2,7 cyklů

b)1/f = 1/500000000 = 2000 ps ; celkový čas = 27000 – 2000 = 54 000 000 ps = 54 ms

1) Strategie pro provádění operace zápisu do paměti

WRITE TROUGH- zápis dat do cache a současně do bloku hlavní paměti-snažší implementace(zápisová buffer) - MISS je jednodušší a levnější, není potřeba zapisovat blok do nižší úrovně

WRITE BACK- zápis pouze do cache - zápis pouze při výměně bloků, výhoda: zápisy omezeny pouze rychlostí zápisu cache, zápis více slov najednou, efektivní pouze zápis celého bloku do paměti

2) Datový hazard RISC

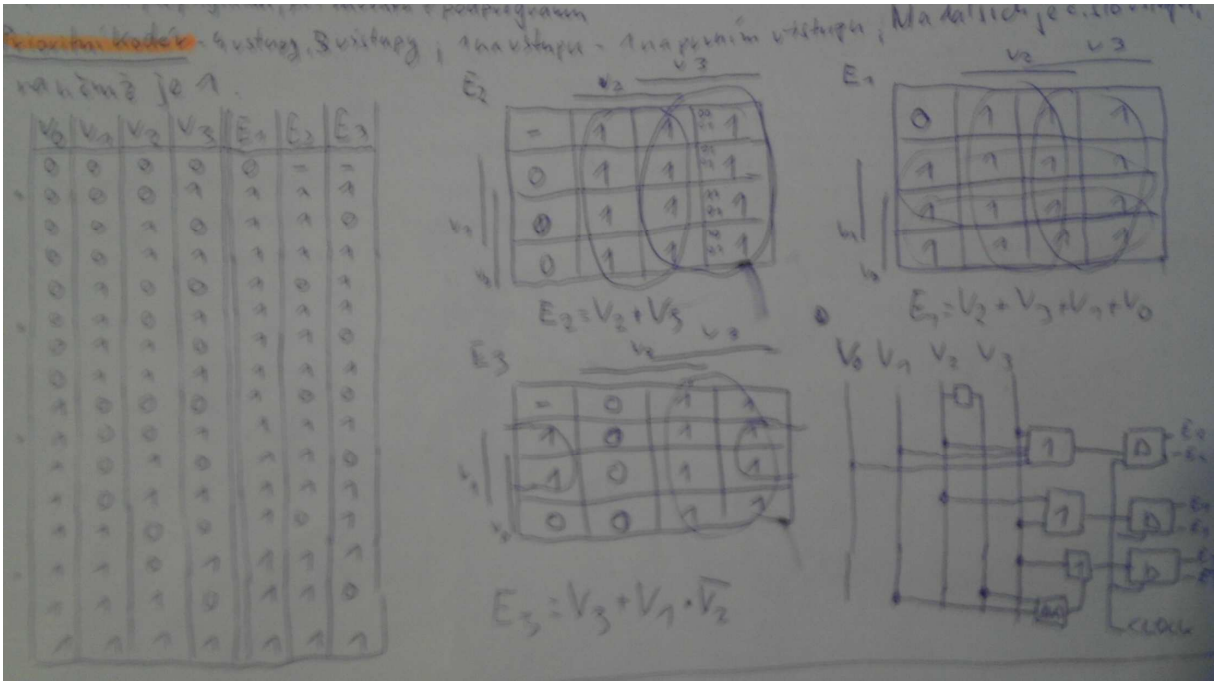
- instrukce je závislá na výsledku předchozí instrukce, která je stále v pipeline(chci počítat s nečím, co se stále počítá)

- řešení - pozastavování vložením tří bublin do pipeline, změna pořadí instrukcí, využití techniky FORWARDING

3) Případy, kdy se mění PC

- zvyšuje se při načtení instrukce, při skoku a větvení, při volání podprogramu, při návratu z podprogramu

4) Navrhnout "prioritní kodér" Zpracovává 4 vstupní signály, vstupy mohou být aktivovány současně.



5) Nejsou-li data v cache nalezena, musí se nejdříve vybrat vhodný blok, který bude zapsán zpět do hl. paměti, a potom může dojít k přenosu požadovaného bloku do cache. Aby se tento proces z hlediska CPU zrychlil, rezervují některé organizace cache volný prostor pro blok, do kterého je nejprve blok do cache přenesen, aby CPU mohl pokračovat v činnosti, a teprve pak je následně provedeno opět uvolnění místa. Uveďte, pro které vám známé organizace cache má toto opatření smysl. Zdůvodněte.

???shit

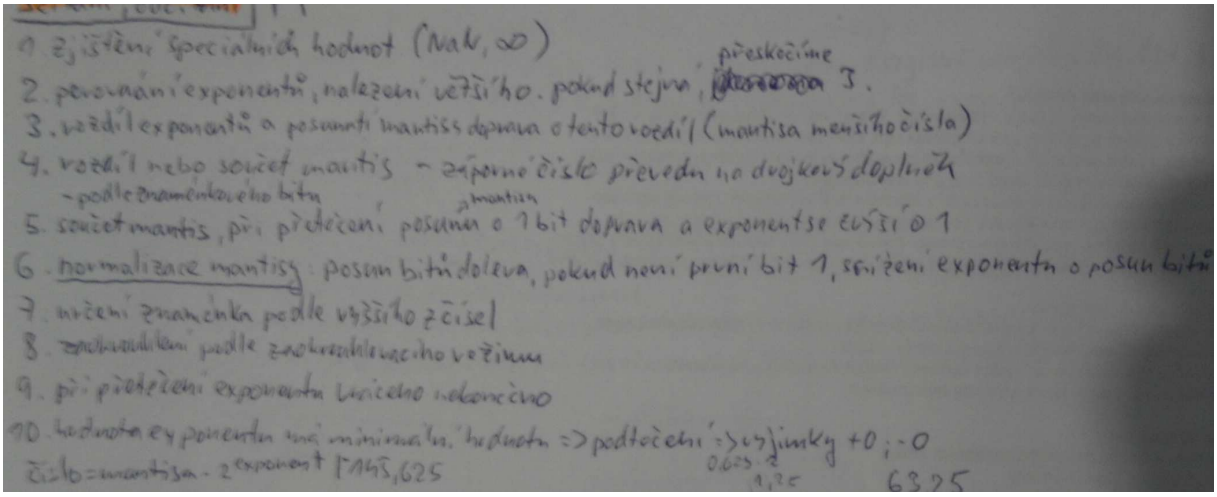
6) Popsat kroky při provádění samotné instrukce volání procedury

- PRED VOLÁNÍ FUNKCE - volající předá argumenty ($a0-$a3;zbylé na stack); uloží registry volajícího ($a0-$a3;$t0-$t9) a provede JAL - skok na proceduru a uložení návratové adresy

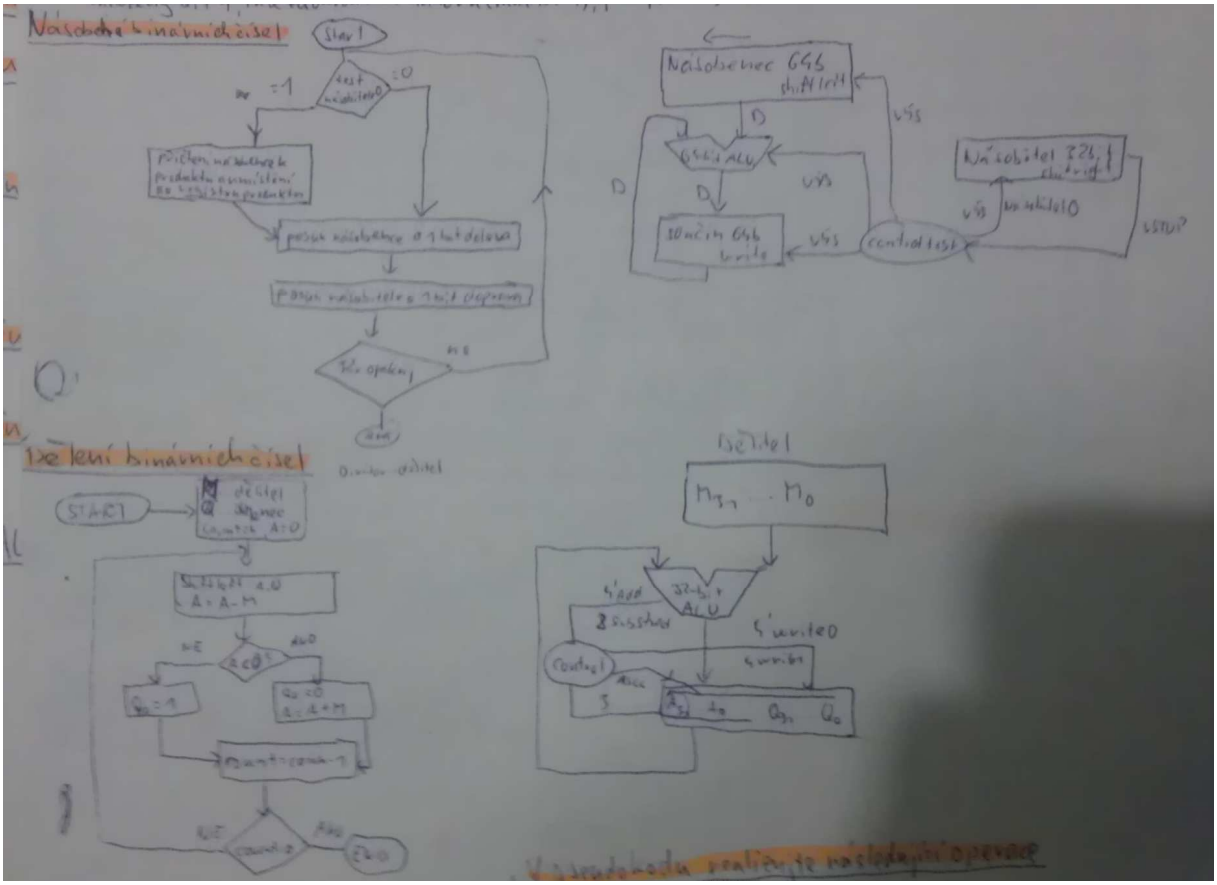
- PŘED ZAHÁJENÍM VÝPOČTU - alokuje se paměť pro frame ($sp=$sp-fsize), uložení ukládaných registrů volaného ($s0-$s7;$fp;$ra)

- PŘED NÁVRATEM DO VOLAJÍCÍHO - uložení funkční hodnoty do registru $r0, obnovení všech registrů funkce, pop stack frame + obnova $fp, návrat na adresu v $ra

7) Jakým způsobem se sčítají čísla v pohyblivé řádové čárce. Podrobně popsat nejlépe načrtnout algoritmus



8) Algoritmus násobení binárních čísel - diagram a operační jednotka



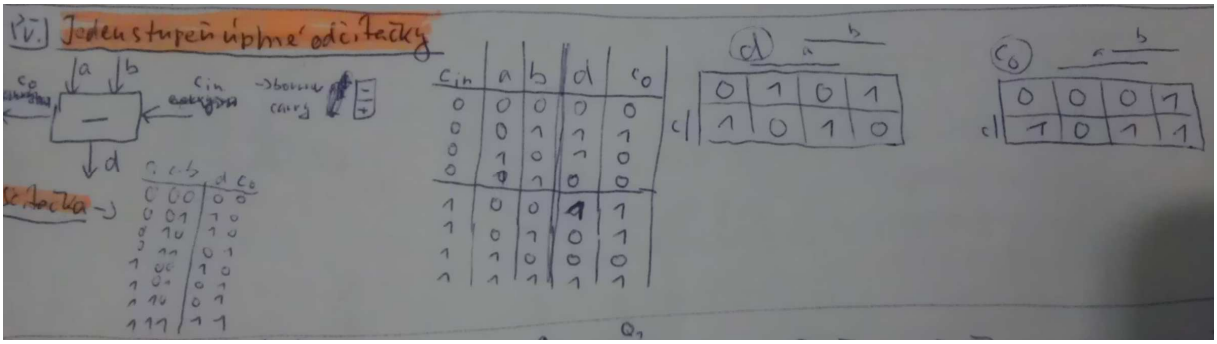
9) for(int i=0; i<4000;i++), for(int j=0; j<8; j++), a[i][j] = b[j][0] + a[j][i]

Kolik 32-bit. čísel typu int může být uloženo v jedné 32 bytové řádce cache? Které přístupy tohoto úseku programu vykazují prostorovou lokalitu?

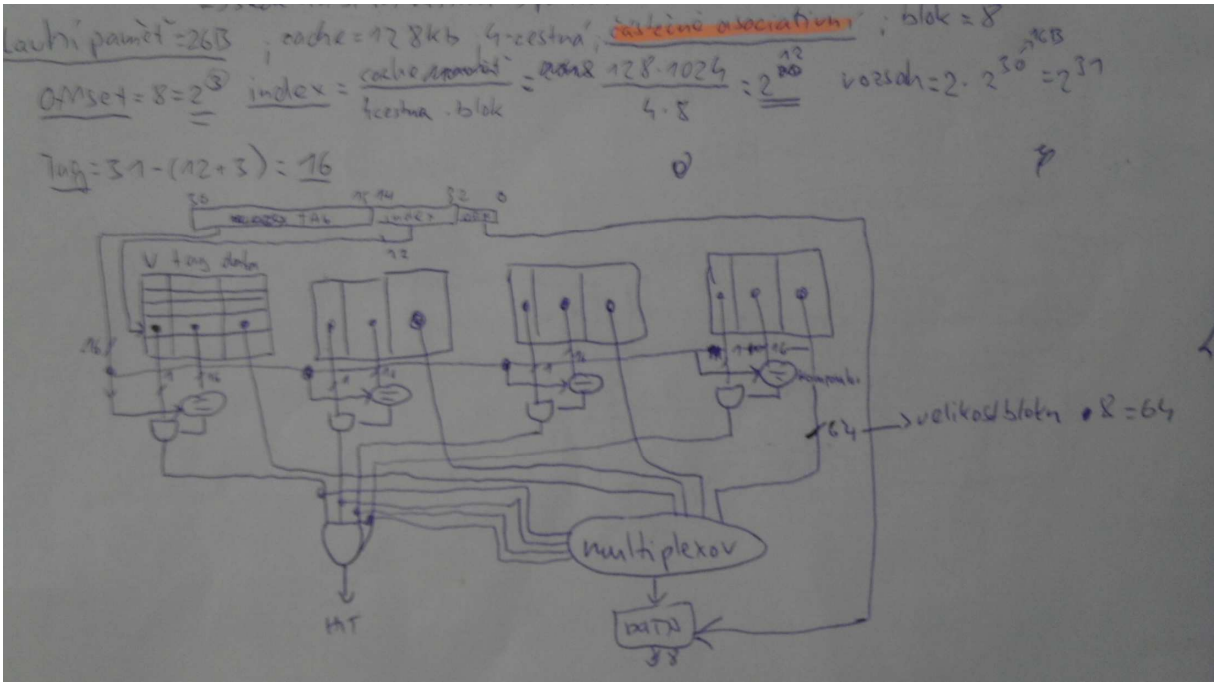
- v jedné řádce může být teoreticky uloženo 8 čísel, ale vejde se jich tam jen 7. lokalita A[i][j]

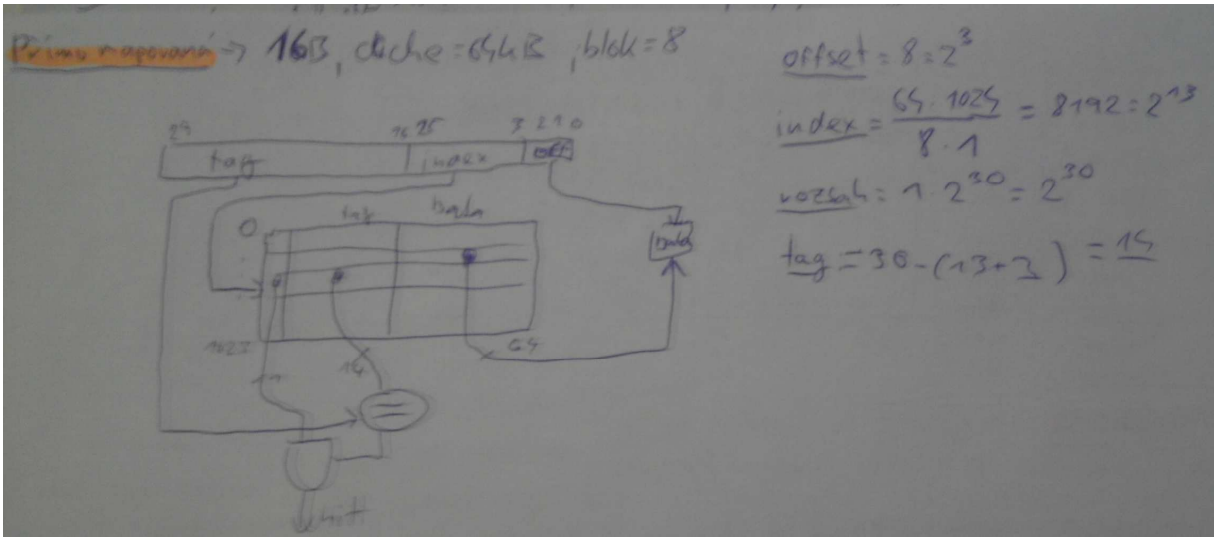
---

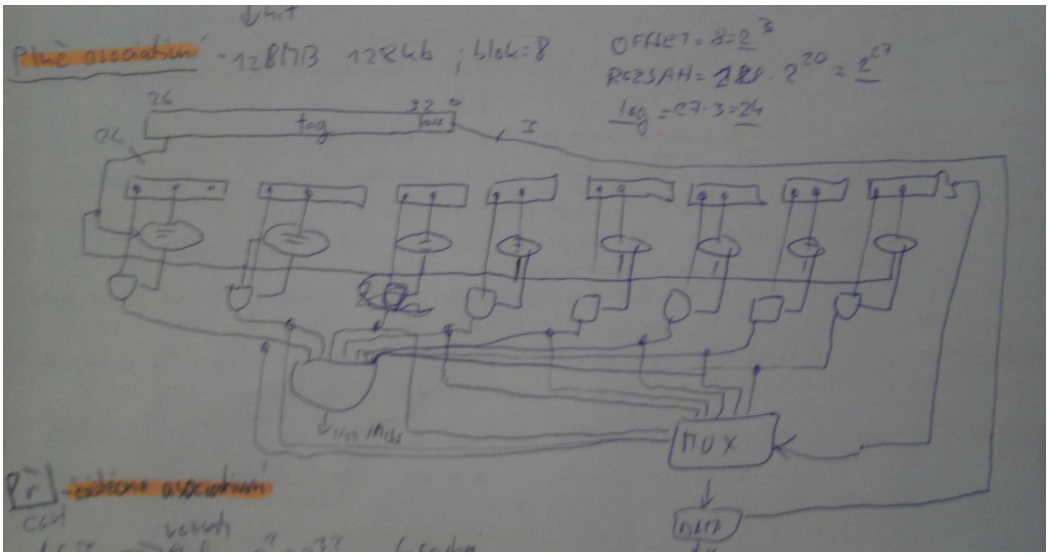
2) Jeden stupeň úplné odčítačky(signály ai, bi, ci) D=A-B



3) HDD 4GB cache 12kB paměti, 4 cestná částečně asociativní paměť. Velikost bloku 16B. Nejmenší adresovatelný blok jeden WORD 2B. Nakreslit a popsat výběrový mechanismus při operaci čtení.







4) Základní charakteristika intstrukčního souboru RISC. Organizace přístupu do ram, jak vypadá formát instrukce, přenos mezi CPU a HDD.Jak se tvoří adresa která je použita pro výběr (uložení) dat.

- lze chápat jako interface mezi HW a SW - abstraktní forma HW

- odděluje implementační složitost SW

- load/store, výpočetní, skoky a větvení, pohyblivá čárka, memory management, speciální

- tři formáty instrukcí - každá 32b. data v paměti pouze přes load-store

5) Princip urychlení práce paralelní binární sčítačky. Schéma pro šířku n bitů.

- máme n-bitovou sčítačku, vytvořenou z obvodů 1-bit úplné, celým obvodem se šíří signál přenosu - zpomaluje činnost

- pro urychlení může při určitých kombinacích vstupů říct, jestli bude generovat přenos nebo předávat signál – drahé, lepší použít několik CLA sčítaček

6) Do jaké kategorie patří řadič CPU. Funkci programového řadiče.

- řídí činnost, je to koneční automat - stav - generování řídících signálů, hrany - podmínky přechodu

- lze implementovat v HW - ROM

1.Instrukční soubor obsahuje instrukce, které mají délku 32bitů. Velikost pamětového adresního prostoru počítače je 230 bytů. Operační kod zabírá 6bitů. Uveďte různé způsoby jak zajistit procesoru přístup k datům, když na adres.instrukci zbývá méně než 27 bitů

- buď uložíme pointer na register, ve kterém bude instrukce uložena (do 32bitů se vejde)

- nebo uložíme pointer na adresu paměti, ve kterém bude instrukce uložena

2.Určete metriky pro hodnocení výkonnosti PC s využitím benchmarků

- čas - Doba CPU -> provedení programu. Hodinová frekv., počet instrukcí -> získává se obtížně (pomocí simulátorů nebo HW čítačů) – závisí na architektuře; nejtěžší je získat CPI-> střední hodnota počtu cyklů na instrukci, ovlivněno všemi atributy návrhu PC.

- výkon – kapacita - spolehlivostní - aby nedošlo k selhání

3.Řídící hazardy, a jakým způsobem se u procesoru třídy RISC omezuje jejich vliv

- ŘH - skoková instrukce pozastavuje zpracování dalších instrukcí

- provedený skok vede až na 3 bubliny v pipeline, to vede k pozastavení,CPU neví, jakou instrukci má načíst

- řešení - opoždění instrukcí větvení, statická a dynamické predikce

4.Navrhněte jednoduchý sekvenčně synchronní automat – 3bitový vratný čítač intuitivní, a standardní – stejný jako 2b

5.Popište podrobně jednotlivé kroky, které procesor vykonává při provádění instrukce podmíněného skoku. Začněte přípravou na... na něco :-(

6.Popište základní registrové sady RISC (nikoliv vlastního instruk.soboru!) Hlavně ty, které maximálně ovlivní výkon procesoru. Nepopisujte urč.jednotlivé registry ani instrukč.formáty. Zajímavé jsou pouze vlastnosti tohoto funkčního bloku

-omezen počet registrů - 32 x 32 bit, pojmenování $t0 nebo číslem $8 , $t -