1. Co je příčinou fyzikálních limitů Mooreova zákona…
   - Předpověděl, že hustota tranzistorů polovodičového čipu se zdvojnásobí každých 18měsíců
   - Limity vlivem nutných izolačních vzdáleností (už se nedá zmenšovat)
   - Kvantové tunelové efekty => crosstalk
   - Řešení zákona jsou kvantové počítače
2. Diagram adresního režimu pro výpočet adresy cíle podmíněného skoku.

Podmíněné skoky

- Instrukce rozhodování
- Skok při rovnosti
  - **beq register1, register2, *destination_address***
- Skok při nerovnosti
  - **bne register1, register2, *destination_address***
- Příklad: beq $s3, $s4, 20

6 bitů	5 bitů	5 bitů	16 bitů
4	19	20	5
000100	10011	10100	0000 0000 0000 0101

ZS 2013

Návěští

- Není třeba vypočítávat adresu cíle skoku

```
      if (i = = j) go to L1;
      f = g + h;
L1:   f = f – i;
```

```
f => $s0
g => $s1
h => $s2
i => $s3
j => $s4
```

```
     (4000) beq $s3, $s4, L1    # if i equals j go to L1
     (4004) add $s0, $s1, $s2   # f = g + h
L1: (4008) sub $s0, $s0, $s3    # f = f - i
```

L1 odpovídá adrese instrukce odčítání

ZS 2013

Multicyklová jednotka: Podmíněný skok

Krok 1: Načtení instrukce // Uložena do IR // Výpočet PC + 4

Krok 2: Dekódování instrukce: pole opcode, rs, rt, offset

Načtení dat: rs a rt určují adresy do sady registrů

Výpočet BTA: SignExt, posun pole offset do buffer registru B

ALU určí PC, offset => BTA

Krok 3: Operace **ALU** (ALUsrcA, ALUsrcB, ALUop) = komparace podle výstupu z ALU (výsledek Zero) dojde k výběru BTA nebo PC+4

CPI pro podmíněný skok = 3 cykly

Pozn.: BTA … adresa cíle instrukce větvení

ZS 2013 16

Provádění instrukce beq

1. Čtení instrukce, např. beq $at, $0, offset z paměti.
2. Čtení zdrojových registrů, $at a $0, z registrového souboru.
3. Odečtením v ALU se obsahy porovnají.
4. Je-li výsledek rozdílu 0 (signál ALU **zero**), zdrojové operandy byly stejné a PC musí být naplněn cílovou adresou PC + 4 + (offset x 4).
5. V opačném případě se skok nekoná a PC je pouze inkrementován na PC + 4 a čte se následující instrukce.

ZS 2013 UPA 24

3. Proč jsou instrukce MIPS lw/sw zařazeny do I-formátu? Jaké to přináší výhody pro návrh MIPS ISA?
   - Lw/sw jsou jediné instrukce pro přístup datům v paměti

Formáty instrukcí

	6 bits	5 bits	5 bits	5 bits	5 bits	6 bits
R:	op	rs	rt	rd	shamt	funct
I:	op	rs	rt	address / immediate		
J:	op	target address				

op: základní operace (opcode)
rs: první operand - registr
rt: druhý operand - registr
rd: registr pro uložení výsledku
shamt: délka posuvu
funct: vybírá specifickou variantu operačního kódu (funkční kód)
address: offset pro instrukce typu load/store (+/-2^{15})
immediate: konstanty pro instrukce s přímými operandy („immediate op.“)

ZS 2013

4. Popište kroky, které procesor vykoná při provádění instrukce návratu z podprogramu.
   - Podprogram před návratem řízení do volají funkce obvykle provede soubor příkazů, které jsou opakem kroků při vstupu do podprogramu. Obnoví se uložené hodnoty registrů, změní se hodnota ukazatele (tedy dealokují se lokální proměnné) a pak se zavolá instrukce návratu, která nahraje ze zásobníku návratovou adresu do instrukčního ukazatele.

5. K čemu se využívají příznaky modifikace a příznak platnosti příslušného bloku cache paměti. Kdy a z jakých důvodů se nastavují.

Nutnost použití příznaku platnosti

- Vedle pole *tag* je třeba zaznamenat, zda blok v cache obsahuje platnou informaci.
  - Při startu procesoru bývá cache naplněna náhodnými daty.
  - I po chvíli činnosti, nemusí být obsah všech položek v cache platný.
- Tento problém lze jednoduše odstranit přidáním bitu platnosti (valid bit) ke každému bloku. Není –li tento bit nastaven, nemůže být blok „nalezen“ (nenastane hit).

Implementace LRU

- U dvoucestné částečně asociativní cache je sledování jednoduché.
  - Ke každému vstupnímu bodu se přidá jeden bit. Kdykoliv je blok referencován, je příslušný bit daného bloku nastaven a odpovídající bit druhého bloku nulován.
  - Nastane-li výpadek a musí dojít k výměně bloků, dojde k ní u bloku, jehož bit je vynulovaný. Bit nového bloku je pak nastaven.
- Situace je poněkud složitější, obsahuje-li skupina více bloků (n = 4, 8).
- Zmíníme se o jiných strategiích výměny. (Jedná se o poměrně složitou problematiku, kterou se nebudeme zabývat do hloubky).

Implementace LRU

- O úspěšnosti VM rozhoduje také algoritmus výměny stránek. Jedním z nich je LRU.
- Předpokládejme, že systém VM používá algoritmus (LRU). OS vytvoří datovou strukturu, obsahující údaje, který proces užívá které fyzické stránky. Tato struktura také udržuje historii o přístupech na stránky, která je pak využita algoritmem LRU.
- Často se jedná jen o aproximaci (skutečný LRU by vyžadoval aktualizaci při každém přístupu).
  - Každá stránka používá referenční bit, který se nastaví po každém přístupu na stránky (R nebo W). OS periodicky nuluje tyto referenční bity – to dovoluje zjistit, na které stránky byl učiněn přístup během určitého časového intervalu.

-Dirty bit pokud je 0, je stránka totožná s její kopiií, v hlavní paměti, a když je 1, byla modifikována (pouze V cachi, slouží to, když vyměňujeme bloky u write-backu, tak zapisuju jen ty data, kde je nastavena 1.

6. Popište alg pro převod čísel se zobrazením v pohyblivé čárce (float) do formy zobrazení v pohyblivé řádové čárce double.

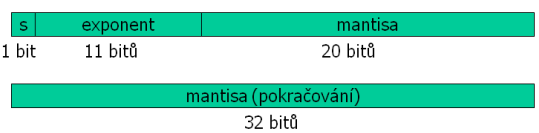
Standardy IEEE 754 (Floating Point)

- IEEE respektoval volby návrhu a doporučil velikost exponentu 8 bitů a 23 bitů pro mantisu (za předpokladu, že délka slova je 32 bitů).
- Tento formát je použit u MIPS a u většiny počítačů po roce 1980 – jedná se dobré kompromisní řešení.

Reprezentované číslo = $(-1)^S \times F \times 2^E$
kde S, F a E jsou pole znaménka, exponentu a mantisy
(1 v poli s znamená zápornou hodnotu čísla)

Dvojitá přesnost

- Aby se bylo možno s těmito případy lépe vyrovnat IEEE 754 standard zahrnuje specifikaci formátu *double precision*, ve které jsou použita dvě slova k zobrazení čísla.
- Exponent je rozšířen na 11 bitů a mantisa na 52 bitů…



- V jazyce C proměnná deklarována jako **double**
- Reprezentuje čísla v rozsahu od nejmenšího 2.0×10^{-308} až po největší 2.0×10^{308}
- Primární výhodou je větší přesnost (52 bitů) (přesnost určuje mantisa !)

Příklad: dekódování IEEE 754

- Zakódujeme $(-0.75)_{10}$ podle IEEE 754.
- Zápis ve dvojkové soustavě… $(-0.11)_2$.
- Normalizovaná forma… $(-1.1)_2 \times 2^{-1}$.
- Požadovaný tvar…

$$(-1)^S \times (1 + \text{mantisa}) \times 2^{(\text{exponent} - \text{bias})}$$

- Převod do požadovaného tvaru…

$$(-1)^1 \times (1 + .1000\,0000\,0000\,0000\,0000\,0000) \times 2^{(126 - 127)}$$

- Pro jednoduchou přesnost podle IEEE 754…

1	0111 1110	1000 0000 0000 0000 0000 000
S	exponent	mantisa

- Pro dvojnásobnou přesnost podle IEEE 754…

0	0111 1111 110	1000 0000 0000 0000 0000 000
S	exponent	mantisa
0000 0000 0000 0000 0000 0000 0000 0000		

-Od exponentu floatu odečtu 127, k 1023 přičtu výsledek (kdyby to bylo záporný, musím to číslo, co připočítám rozšířit na 11b) a za mantisu přidám 0, aby to bylo 64bitů.

7. 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:

Load 0 - povinný miss - cache je prázdná

0

Load 2 - povinný miss

0
4
1
2

Load 5 - kapacitní miss

3
5
1
2

Load 4 - povinný miss

0
4

Load 3 - kapacitní miss - nevejde se už do cache, vyhodím nejstarší

3
4
1
2

Load 6 - kapacitní miss

3
5
6
2

Load 1 - povinný miss

0
4
1

Load 4 - hit

3
4
1
2

Load 4 - kapacitní miss

3
5
6
4

Load 0 - hit

0
4
1

Load 6 - hit

3
5
6
4

8. 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.

- Zarovnána data chápeme jako data ke kterým přistupujeme po 4.. tj. M[0], M[4],.. M[4n]
- Lw/sw vyžadují zarovnaná data
- procesory RISC používají zarovnané instrukce, že jsou všechny stejně dlouhé a usnadňuje to pipeline, všechny trvají stejně dlouho
- Výhoda: data jsou systematicky umístěny
- Zarovnání dat znamená umístit data v paměti na takovou pozici, která je rovna násobku velikosti slova – bloku dat, se kterými standardně pracuje systém. To zvyšuje výkon celého systému vzhledem ke způsobu, jakým procesor pracuje s pamětí. Pro správné zarovnání dat může být nezbytné vložit za data, která netvoří svojí délkou úplný blok, na jejich konec nějaké nesmyslné bajty. Tomu se právě říká data structure padding (nezarovnaná) – „vyplnit to vatou“.

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.

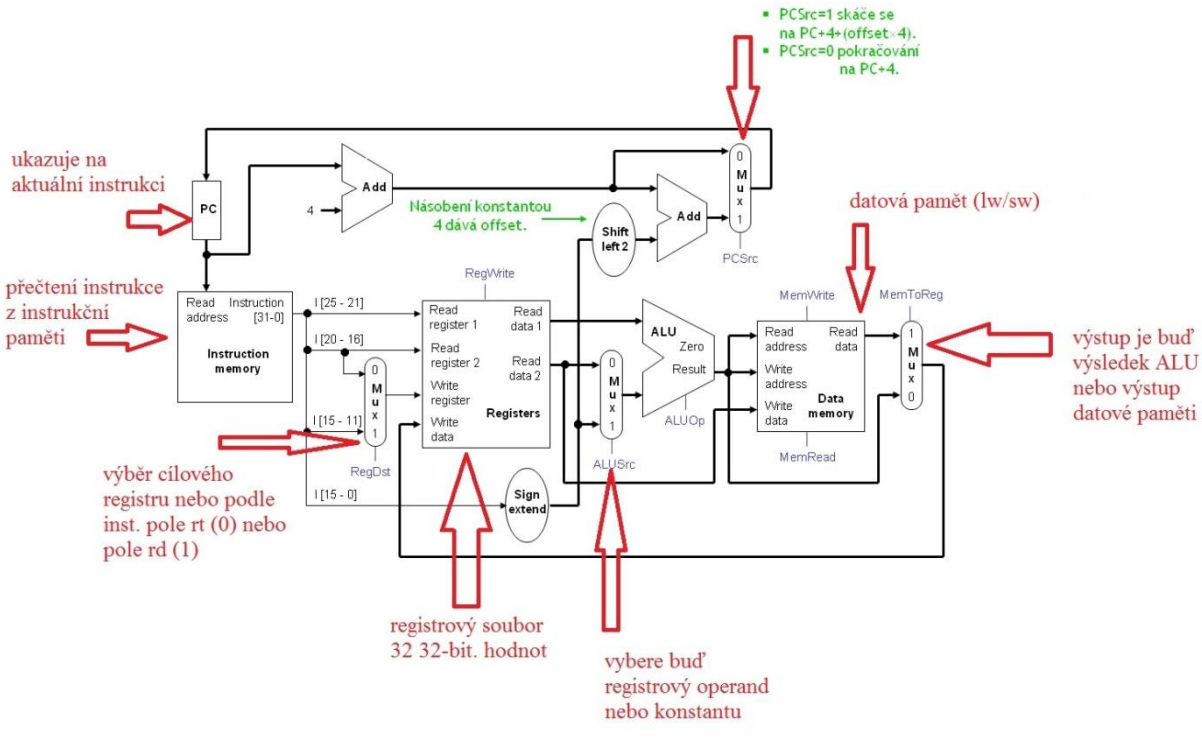
   a) Jaká je střední hodnota parametru CPI pro program P?

   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?

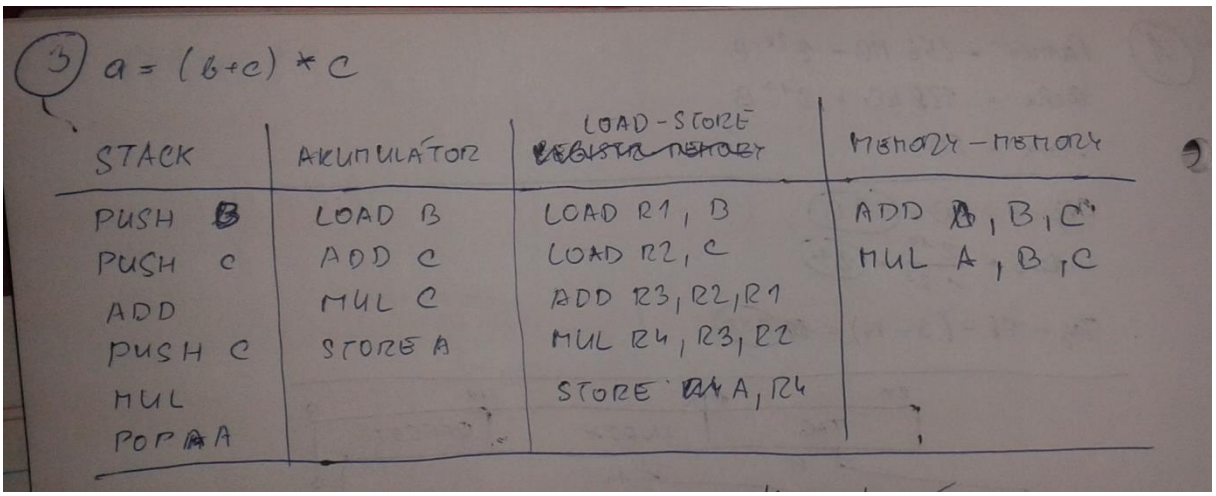
a) 1*0,2+2*0,3+3*0,1+4*0,4=2,7

b) 1/f = 1/500 000000 = 2ns ,
(1*0,2*10000+2*0,3*10000+3*0,1*10000+4*0,4*10000)*2=54000ns a 27000cyklů

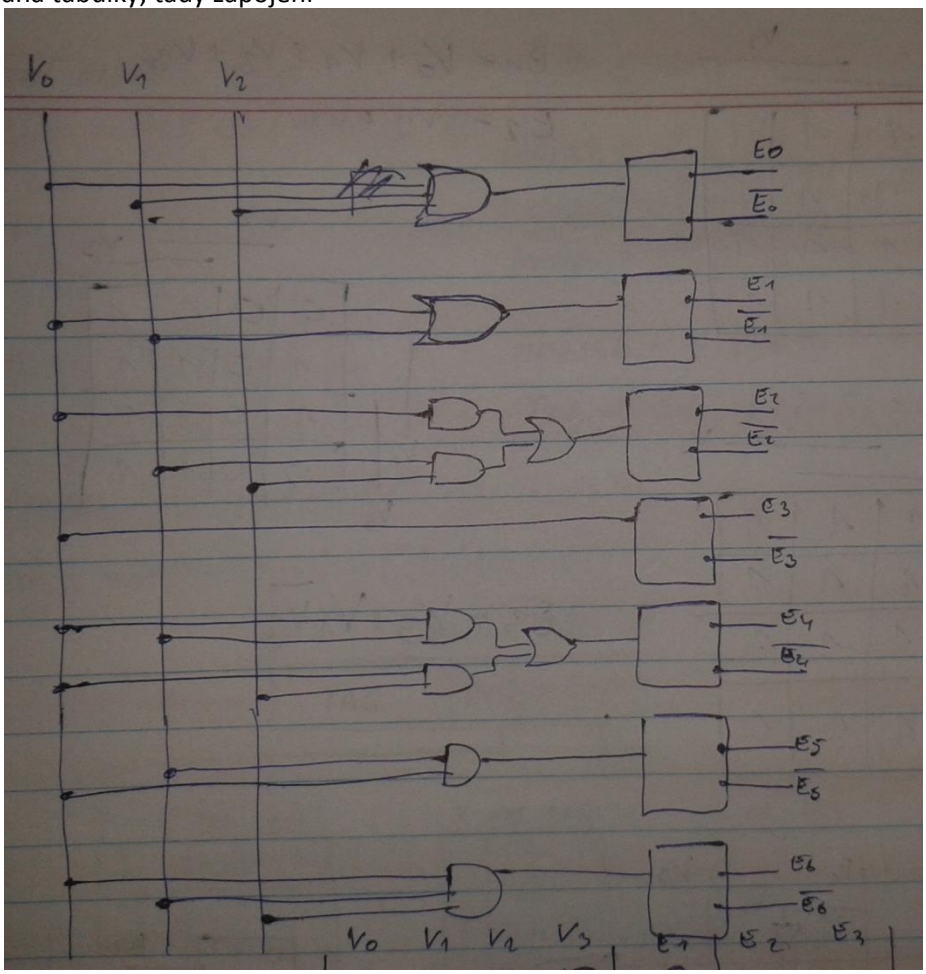
10. Obrázek



11. Popište, jakým způsobem se dělí čísla v pohyblivé řádové čárce.
    Nejdřív se vydělí mantisy i přes to, že mají jiný exp. Pak se exponenty od sebe odečtou a výsledek se normalizuje. Ještě je třeba dát si pozor na znaménko - jestliže se dělila dvě záporná čísla, tak je výsledek kladný, ++ -> +; +- nebo -+ dá -.

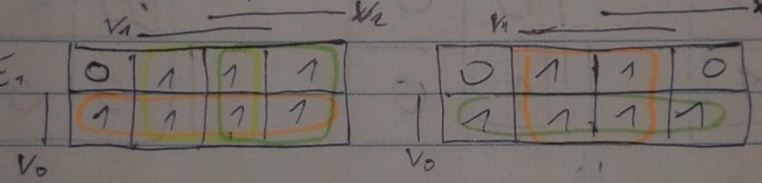
12. 

13. Další strana tabulky, tady zapojení



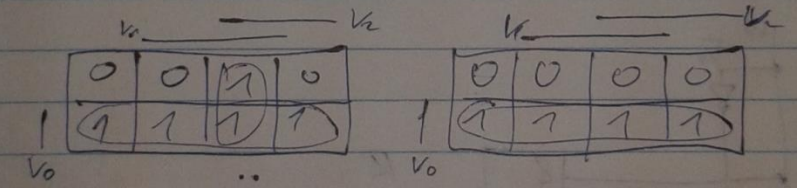
3 vstupy, 7 výstupů – Počet aktivních vstupů je roven binárnímu číslu na výstupu.

	V_0	V_1	V_2	E_0	E_1	E_2	E_3	E_4	E_5	E_6
0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	0	0	0
2	0	1	0	1	1	0	0	0	0	0
3	0	1	1	1	1	1	0	0	0	0
4	1	0	0	1	1	1	1	0	0	0
5	1	0	1	1	1	1	1	1	0	0
6	1	1	0	1	1	1	1	1	1	0
7	1	1	1	1	1	1	1	1	1	1



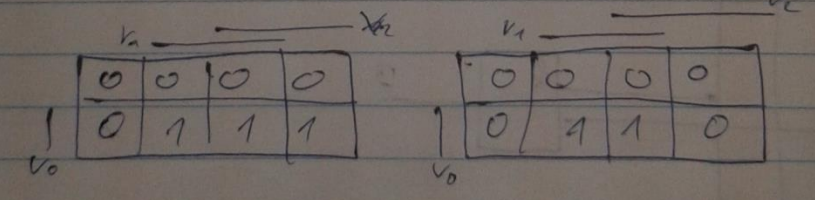
$E_0 = V_0 + V_1 + V_2$

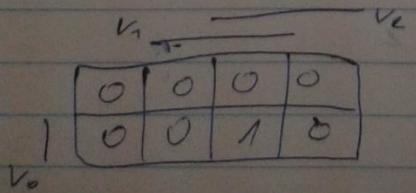
$E_1 = V_0 + V_1$



$E_2 = V_0 + V_1 V_2$

$E_3 = V_0$



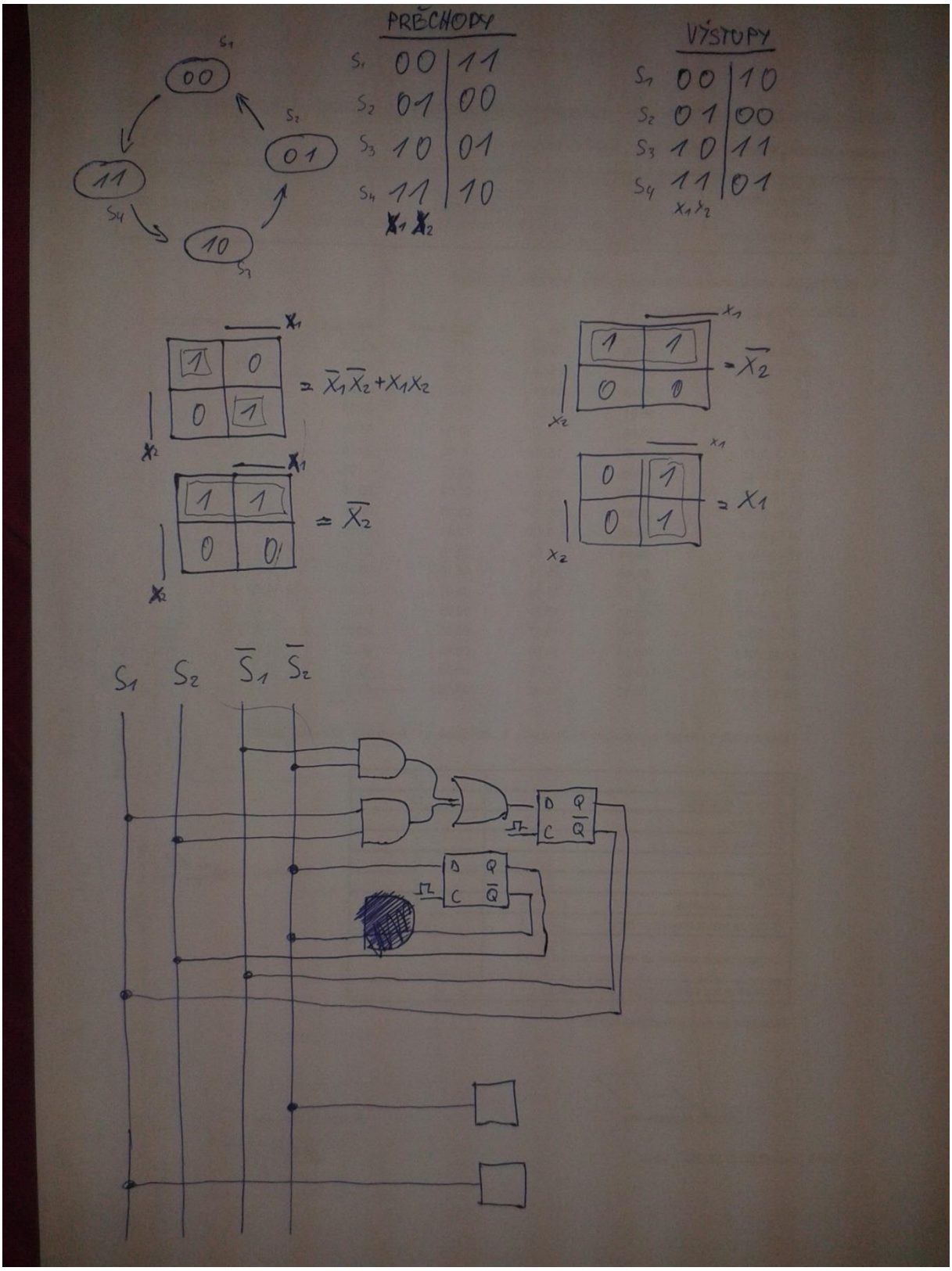


$E_4 = V_1 V_0 + V_2 V_0$

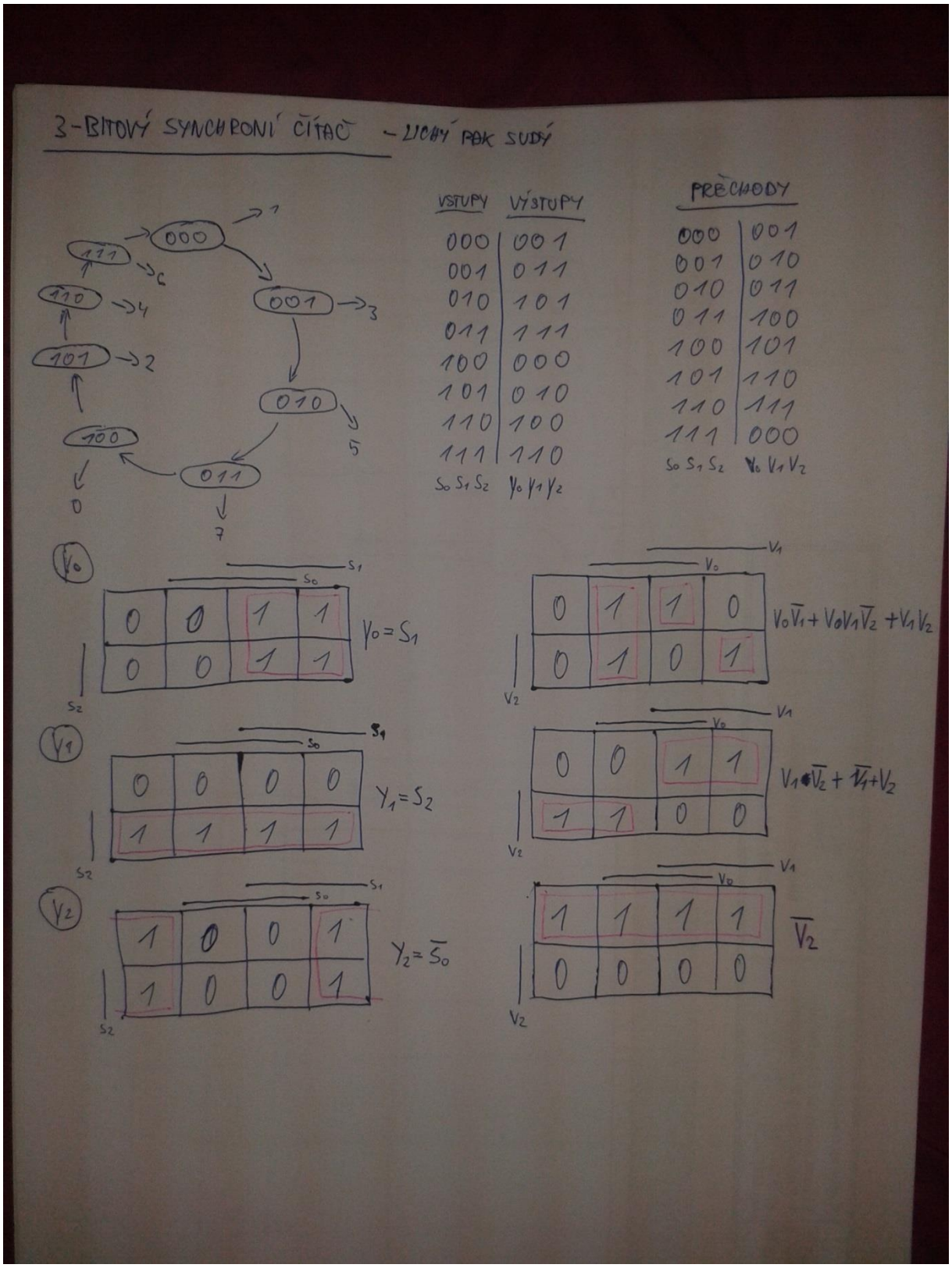
$E_5 = V_1 V_0$

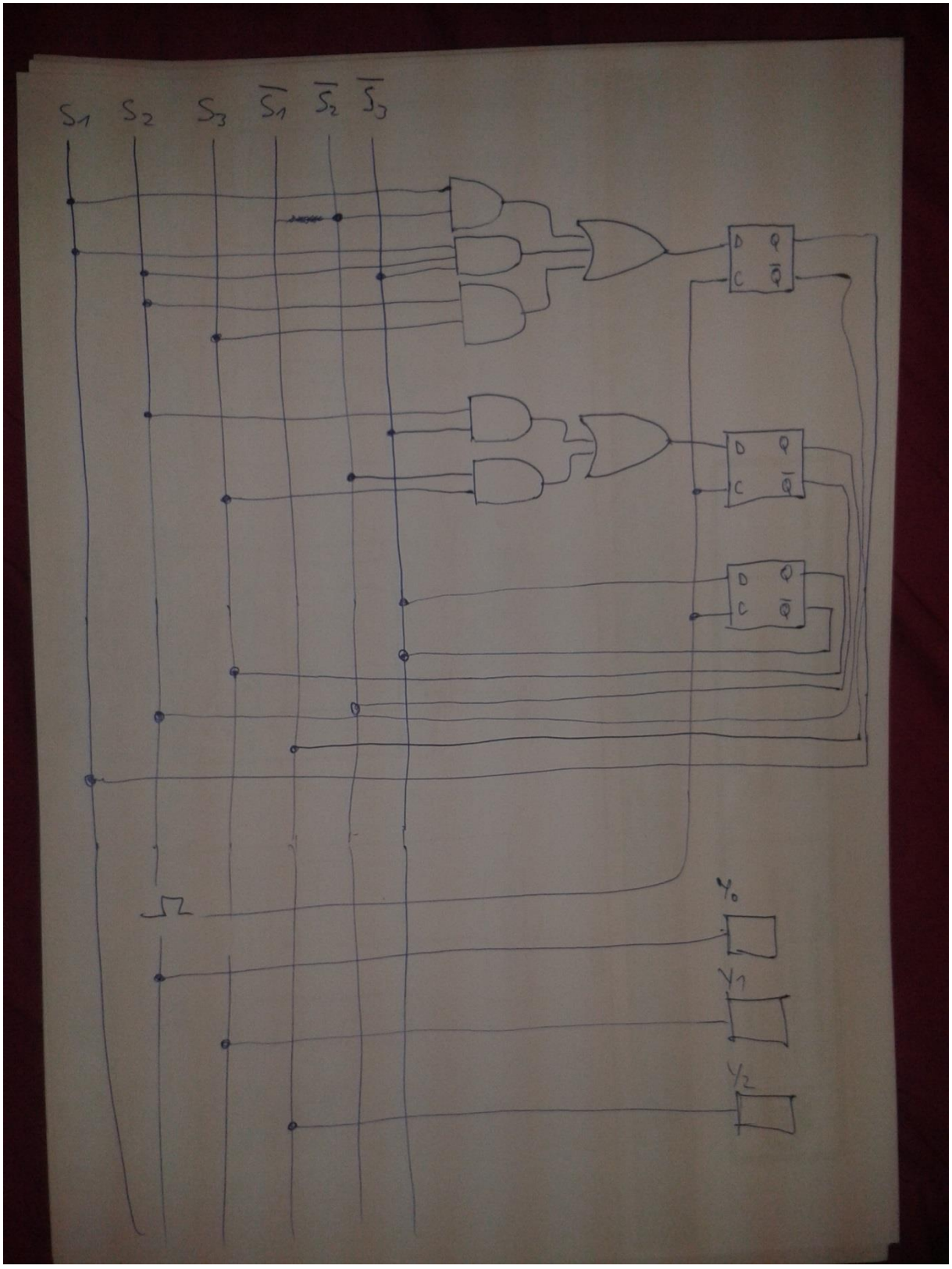
$E_6 = V_0 V_1 V_2$

14. Binární reverzní čítač

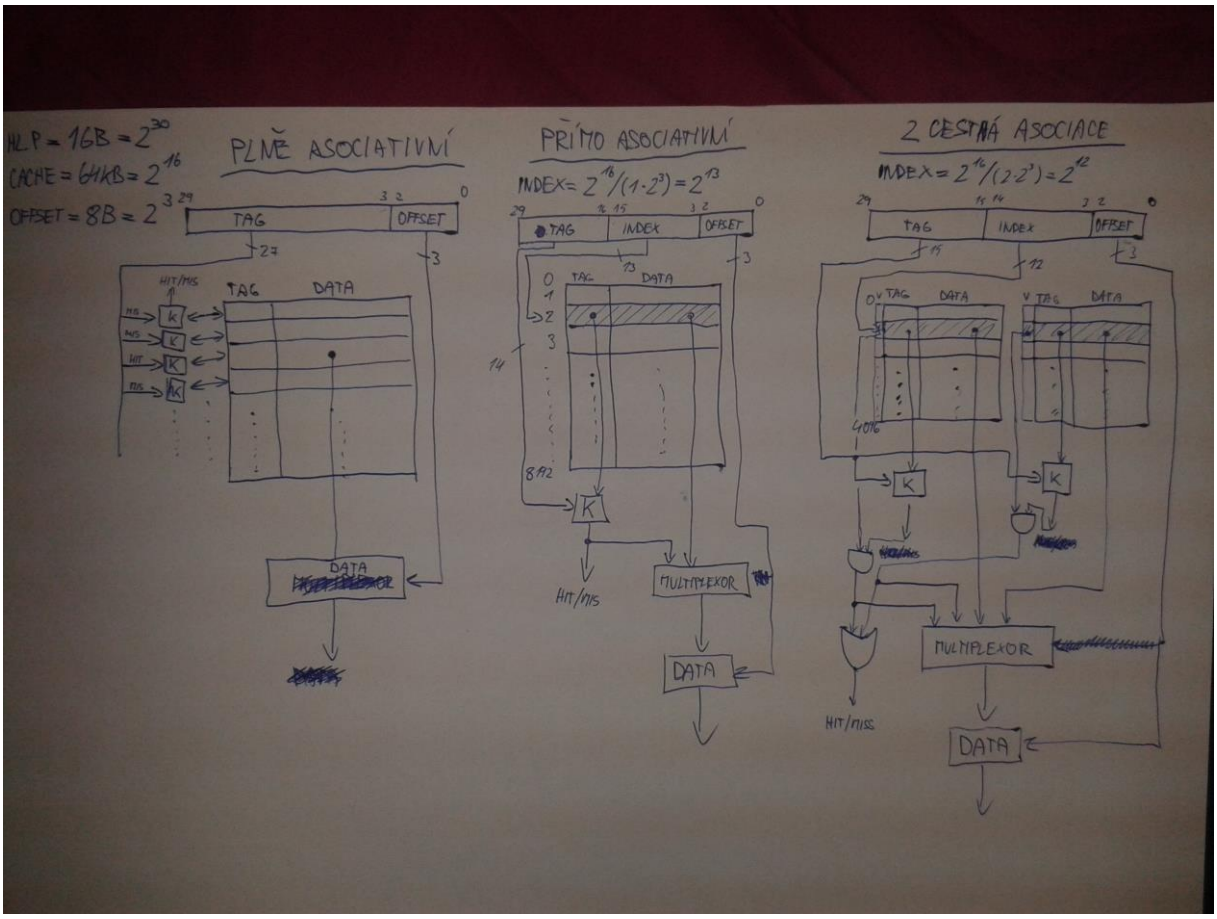


15. 3bitový synchronní čítač

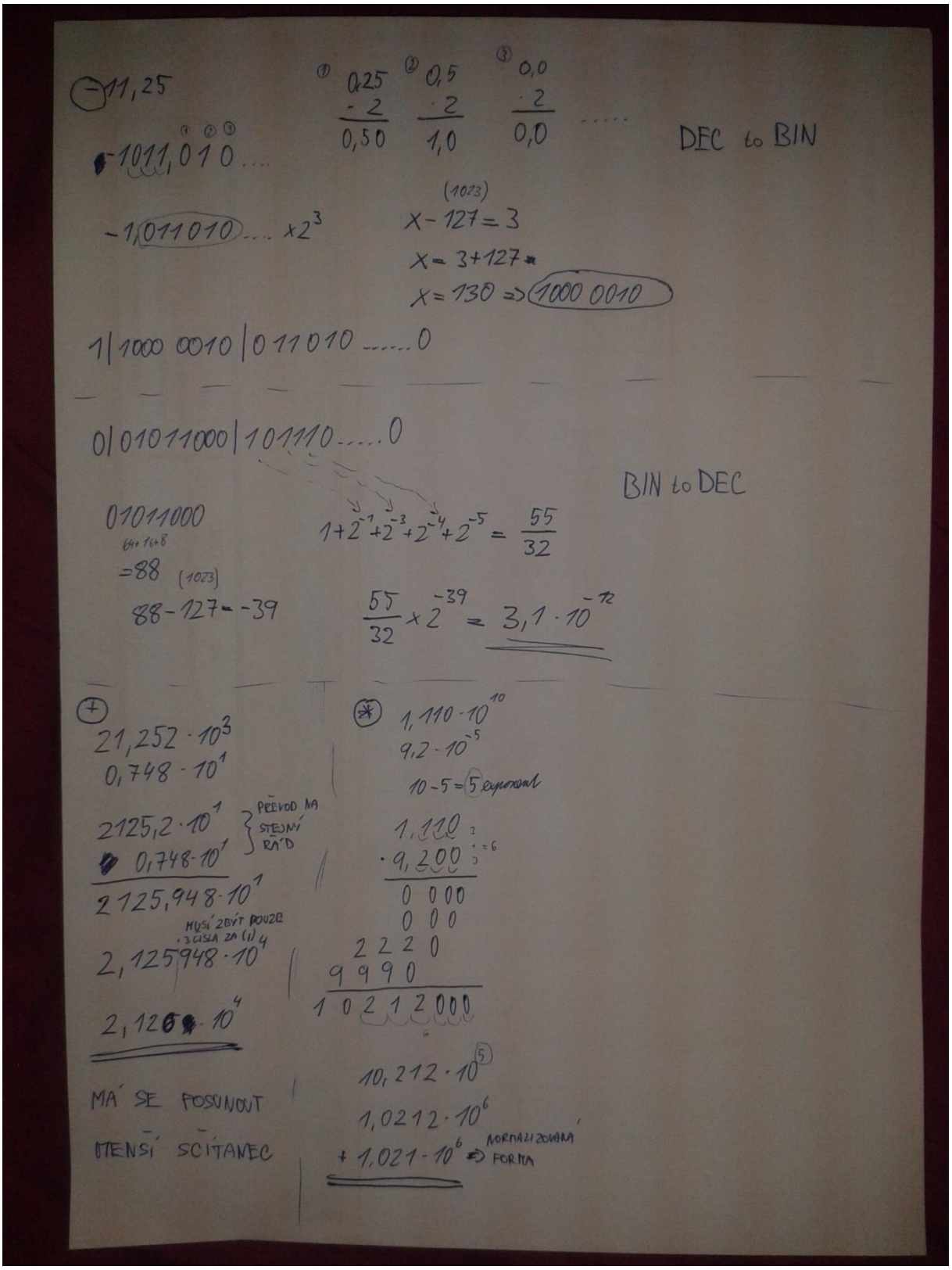



S1
S2
S3
S̄1
S̄2
S̄3
D
C
Q
Q̄
Y0
Y1
Y2

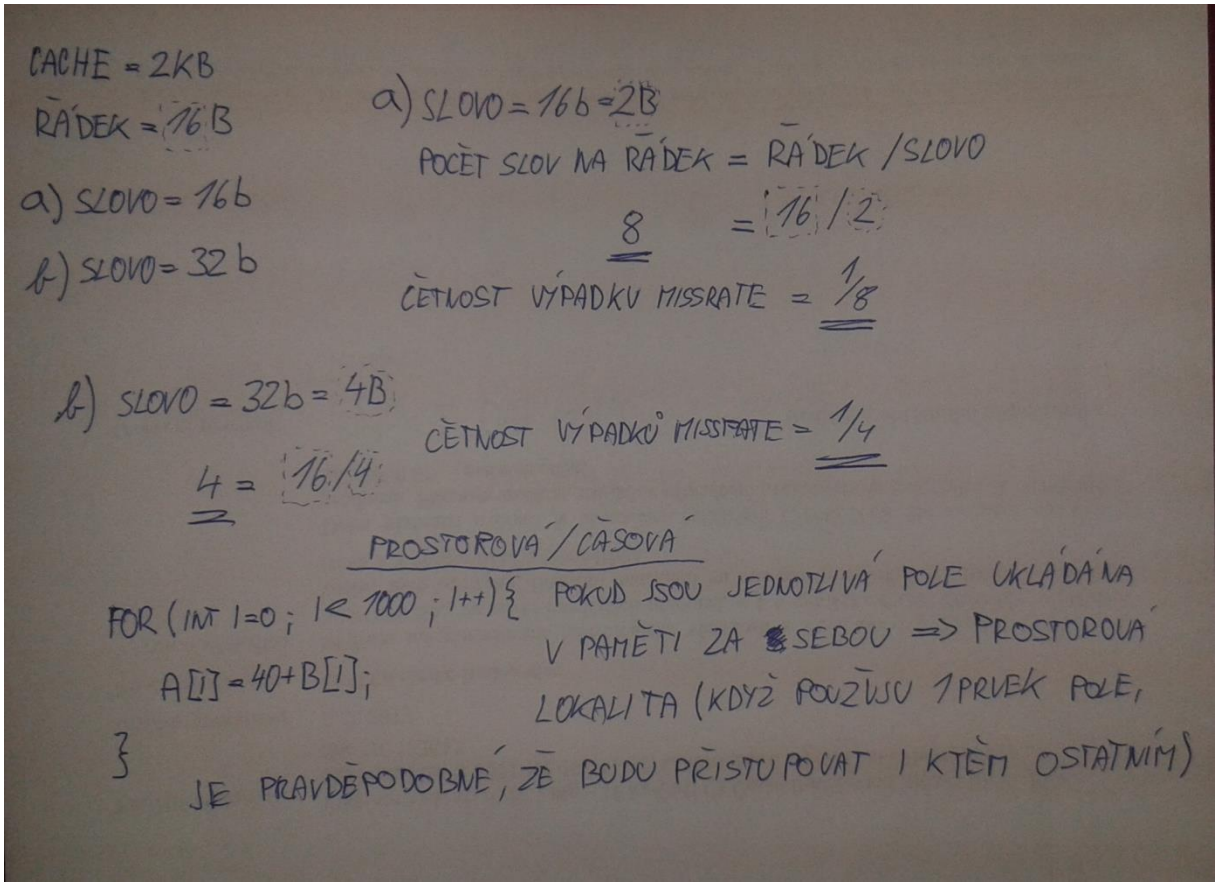
16. Cache


HLP = 1GB = 2^30
CACHE = 64KB = 2^16
OFFSET = 8B = 2^3
PLNĚ ASOCIATIVNÍ
TAG
OFFSET
HIT/MIS
DATA
K
MULTIPLEXOR
PŘÍMO ASOCIATIVNÍ
INDEX = 2^16/(1·2^3) = 2^13
INDEX
2 CESTNÁ ASOCIACE
INDEX = 2^16/(2·2^3) = 2^12
V

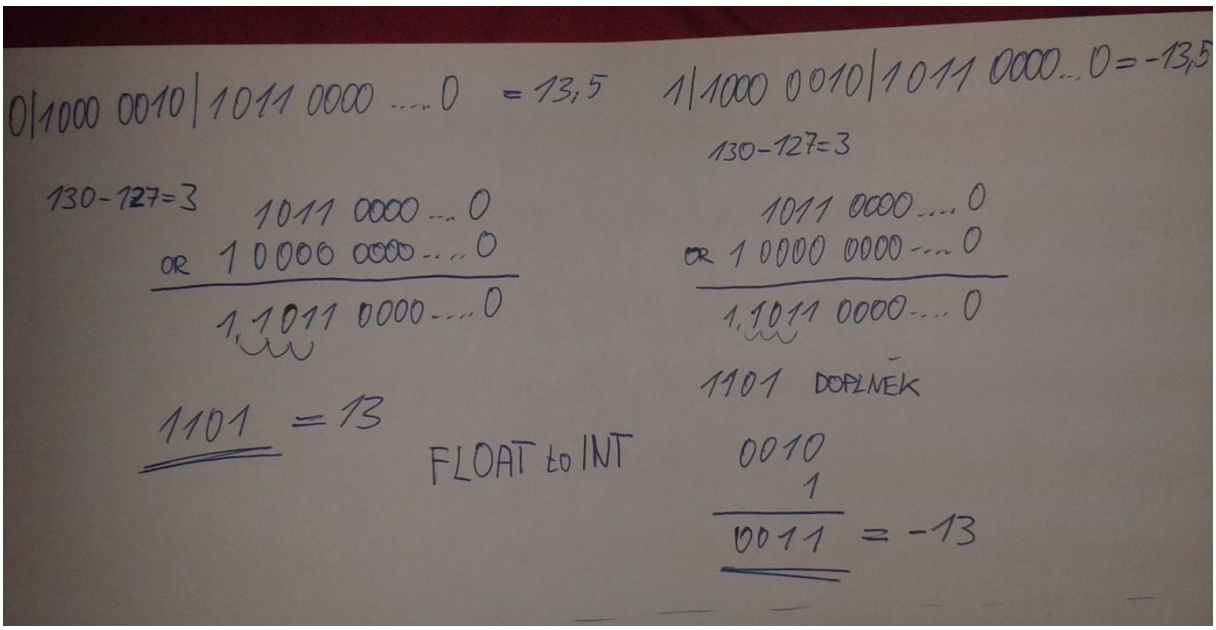
17.


-11,25
-1011,010....
DEC to BIN
0,25 · 2 = 0,50
0,5 · 2 = 1,0
0,0 · 2 = 0,0
-1,011010.... x2^3
(1023)
X - 127 = 3
X = 3+127
X = 130 => 1000 0010
1|1000 0010|011010.....0
0|01011000|101110.....0
BIN to DEC
01011000
64+16+8
=88 (1023)
88-127=-39
1+2^-1+2^-3+2^-4+2^-5 = 55/32
55/32 x 2^-39 = 3,1 · 10^-12
21,252 · 10^3
0,748 · 10^1
2125,2 · 10^1
0,748 · 10^1
PŘEVOD NA STEJNÝ ŘÁD
2125,948 · 10^1
MUSÍ ZBÝT POUZE 3 ČÍSLA ZA (,)
2,125948 · 10^4
2,1260 · 10^4
MÁ SE POSUNOUT MENŠÍ SČÍTANEC
1,110 · 10^10
9,2 · 10^-5
10-5=5 exponent
1,110 · 9,200
0000
000
2220
9990
10212000
10,212 · 10^5
1,0212 · 10^6
+ 1,021 · 10^6
NORMALIZOVANÁ FORMA

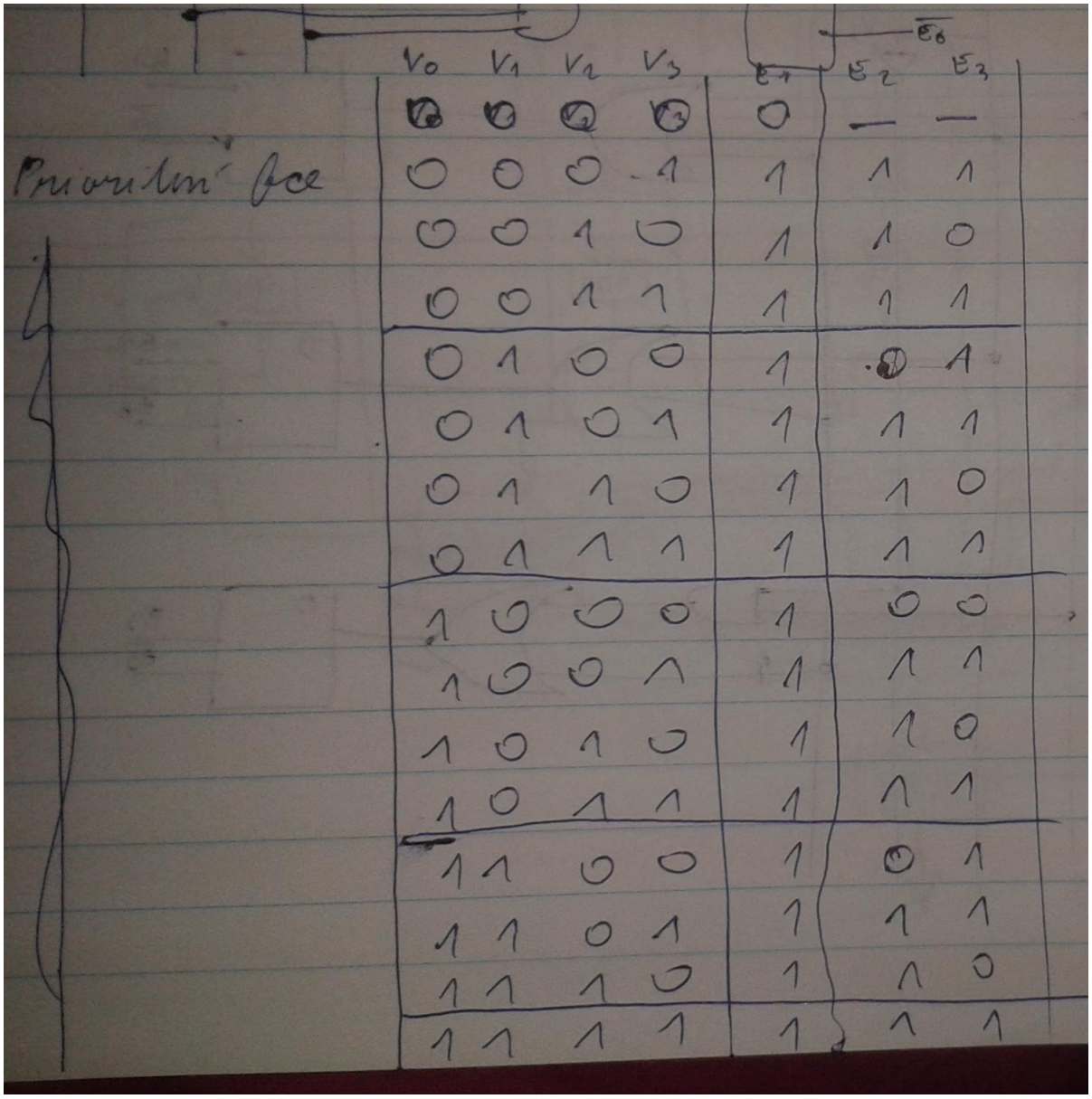
18. četnost



19. float -> int



20. Prioritní fce



E_1

V_2 \ V_3	00	01	11	10
V_1 0	0	1	1	1
1	1	1	1	1
1	1	1	1	1
V_0 1	1	1	1	1

$E_1 = V_0 + V_1 + V_2 + V_3$

$E_2 = V_2 + V_3$

V_2 \ V_3	00	01	11	10
V_1 0	0	0	1	1
1	0	0	1	1
1	0	0	1	1
V_0 1	0	0	1	1

E_2

V_1 \ V_2	00	01	11	10
V_0 0	0	1	1	1
0	0	1	1	1
0	0	1	1	1
1	0	1	1	1

$E_3 = V_3 + V_1 \overline{V_2}$