

>\$01

# AK47/OiAK 2017

## OPRACOWANIA - Patronik

**2016 termin 0                    ZAD 1,2,3,4,5,6 /2017 NA ZERÓWCE!**

**2015 termin 0                    ZAD 1,2,3,4,5**

**2014 termin 0                    ZAD**

)

**2014 termin 1                    ZAD 1,2,3,4,5**

**2016 termin 1 i 2                ZAD 1,2,3,3,4,4,5,5,6,6 /2017 NA ZERÓWCE!**

**ZROBIONE OK, NIEZROBIONE WYMAGA POPRAWY**

**Dysk Patronusa (wyniki):**

<https://drive.google.com/drive/u/0/folders/0Bxrgd6x3m9Nicng5anM5SE5NTWM>

Slajdy Batonia:

<https://drive.google.com/drive/folders/0Bxrgd6x3m9NiZUIxQ1JORmhWkE?tid=0Bxrgd6x3m9NiUkZQXy1aSDI5YUU>

**TERMINY:**

INEK00003W	Architektura komputerów 2	dr inż. Piotr Patronik	28.06.2017 godz. 15:00 – 17:00 s. 201 C-1	04.07.2017 godz. 15:00 – 17:00 s. 201 C-1
INEK00022W	Organizacja i architektura komputerów	dr inż. Piotr Patronik	29.06.2017 godz. 11:00 – 13:00 s. 201 C-1	06.07.2017 godz. 11:00 – 13:00 s. 201 C-1

**DZIĘKUJĘ PAN PIOTR**

<b>(Koło 13.06.2016, TERMIN 0 -</b>	<b>3</b>
Zadanie 1. 2016 termin 0	5
Zadanie 2. 2016 termin 0	8
Zadanie 3. 2016 termin 0	11
Zadanie 4. 2016 termin 0	15
Zadanie 5. 2016 termin 0	17
Zadanie 6. 2016 termin 0	19
<b>Koło 2015, TERMIN 0</b>	<b>21</b>
Zadanie 1. 2015 termin 0	22
Zadanie 2. 2015 termin 0	23
Zadanie 3. 2015 termin 0	33
Zadanie 4. 2015 termin 0	36
Zadanie 5. 2015 termin 0	39
<b>Koło 2014, TERMIN 0</b>	<b>40</b>
Zadanie 1. 2014 termin 0	41
Zadanie 2. 2014 termin 0	43
Zadanie 3. 2014 termin 0	46
Zadanie 4. 2014 termin 0	49
Zadanie 5. 2014 termin 0	50
<b>Koło 2014, TERMIN 1</b>	<b>51</b>
Zadanie 1. 2014 termin 1	52
Zadanie 2. 2014 termin 1	55
Zadanie 3. 2014 termin 1	57
Zadanie 4. 2014 termin 1	60
Zadanie 5. 2014 termin 1	63
<b>Koło 2016, TERMIN 1 i 2 (bo wiele się powtarza) - to prawdopodobnie będzie na o TERMINIE 2017</b>	<b>64</b>
Zadanie 1. 2016 termin 1 i 2	65
Zadanie 2. 2016 termin 1 i 2	66
Zadanie 3. 2016 termin 1	67
Zadanie 3. 2016 termin 2	68
Zadanie 4. 2016 termin 1	69
Zadanie 4. 2016 termin 2	70
Zadanie 5. 2016 termin 1	71
Zadanie 5. 2016 termin 2	72

Zadanie 6. 2016 termin 1	73
Zadanie 6. 2016 termin 2	76

**(Koło 13.06.2016, TERMIN 0 -  
to prawdopodobnie będzie na 0 TERMINIE 2017 - I BYŁO!)**

**Koło 13.06.2016, TERMIN 0**

**ROZWIĄZANIA:**

**Zadanie 1. 2016 termin 0**

I. (4p) Jest dany procesor o 4-bitowym słowie rozkazowym i poniższym kodowaniem rozkazów, w którym czas wykonania każdej mikrooperacji wynosi 50ps. Zapisać program w postaci mnemoników. Podać czas wykonania 6 rozkazów (zapisanych szesnastkowo):  $h_3, h_1, 0xFE, h_0, h_4$  dwóch przypadkach: (i) procesor jest w pełni sekwencyjny, (ii) mikrooperacje (F, W), (F, E), (D, W) mogą być wykonane równocześnie w potoku.

Kod	Zapis	Operacja	Mikrooperacje
$ii\ v\ 1$	$ld\ \$v,\ %ri$	$ri \leftarrow v$	FDW
$ii\ j\ 0$	$add\ %ri,\ %rj$	$rj \leftarrow ri+rj$	FDREW

Patronik używa naszego numeru indeksu jako liczb odpowiednio  $h_5, h_4, h_3, h_2, h_1, h_0$ . Musimy wziąć 4-bity o które prosi Patronik więc interesują nas  $h_3, h_1, h_0, h_4$  liczby indeksu (tutaj 8461). W poleceniu mamy wykonać operacje na  $h_3, h_1, 0xF, 0xE, h_0, h_4$ . Każdą z liczb rozpisujemy dwójkowo? Bity liczymy od 0;  $h_0=6, h_1=4, h_2=2, h_3=8, h_4=1, h_5=2$   
W przypadku indeksu **218246** będzie to rozpisane kolejno:

Dla  $h_3: 1000$  --- add %r2, %r0 --- FDREW  
 Dla  $h_1: 0100$  --- add %r1, %r0 --- FDREW  
 Dla  $0xF: 1111$  --- Id \$1, %r3 --- FDW  
 Dla  $0xE: 1110$  --- add %r3, %r1 --- FDREW  
 Dla  $h_0: 0110$  --- add %r1, %r1 --- FDREW  
 Dla  $h_4: 0001$  --- Id \$0, %r0 --- FDW

Dla J24 00010002 --- add %r1000, %r1001 -- MNTW  
 Dla ZX2346 0002 --- add %r 23, %123 -- DEW

Dlaczego tak? Wyjaśnienie: //dlaczego? bo słabo kminie

Linia nr 1(add):

1000 - 2 (stad r2)

1000 - 0 (stad r0)

1000 - dlatego operacja add

Linia nr 3(Id):

1111 - 3 (stad r3)

1111 - 1 (stad \$1)

1111 - dlatego operacja Id

1

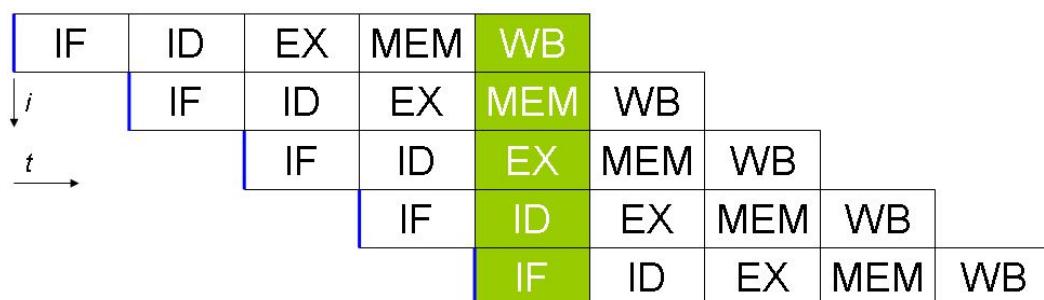
Mamy już rozpisane te 6 rozkazów w postaci mnemoników. Teraz musimy obliczyć czas ich wykonania:

i) w przypadku procesora w pełni sekwencyjnego:

Wiemy że czas wykonania 1 operacji zajmuje 50ps więc w przypadku pierwszego rozkazu będzie to  $5 \times 50\text{ps} = 250\text{ps}$ . Wykonanie tych 6 rozkazów zajmie  $26 \times 50\text{ps} = \mathbf{1300\text{ps}}$  (liczba wszystkich mikrooperacji \* czas wykonania pojedynczej mikrooperacji).

ii) w przypadku procesora gdzie mikrooperacje (F,W), (F,E), (D,W) mogą być wykonywane równocześnie w potoku

// Według Patronika: równocześnie w potoku wykonujemy operacje z RÓŻNYCH rozkazów //no



chyba oczywiste skoro są rozpisane operacje, które mogą być wykonywane równocześnie

// #Konsultacje: Jeżeli ktoś zrobił wszystko dobrze, a nie miał rysunku z potokami dla drugiego przypadku dostawał max 1 pkt.

#pytanie czemu na dole najpierw jest EXecute a potem Memory?

Dla h3: 1000	--- add %r2, %r0	--- FDREW
Dla h1: 0100	--- add %r1, %r0	--- FDREW
Dla 0xF: 1111	--- Id \$1, %r3	--- FDW
Dla 0xE: 1110	--- add %r3, %r1	--- FDREW
Dla h0: 0110	--- add %r1, %r1	--- FDREW
Dla h4: 0001	--- Id \$0, %r0	--- FDW

(FE) (FW) (DW)

Ludzie zostawcie to... Nie bazgrajcie, nie griejujcie.

Rozwiążanie do numeru indeksu 218246 : //DO SPRAWDZENIA - (F,W), (F,E), (D,W)

Czas:

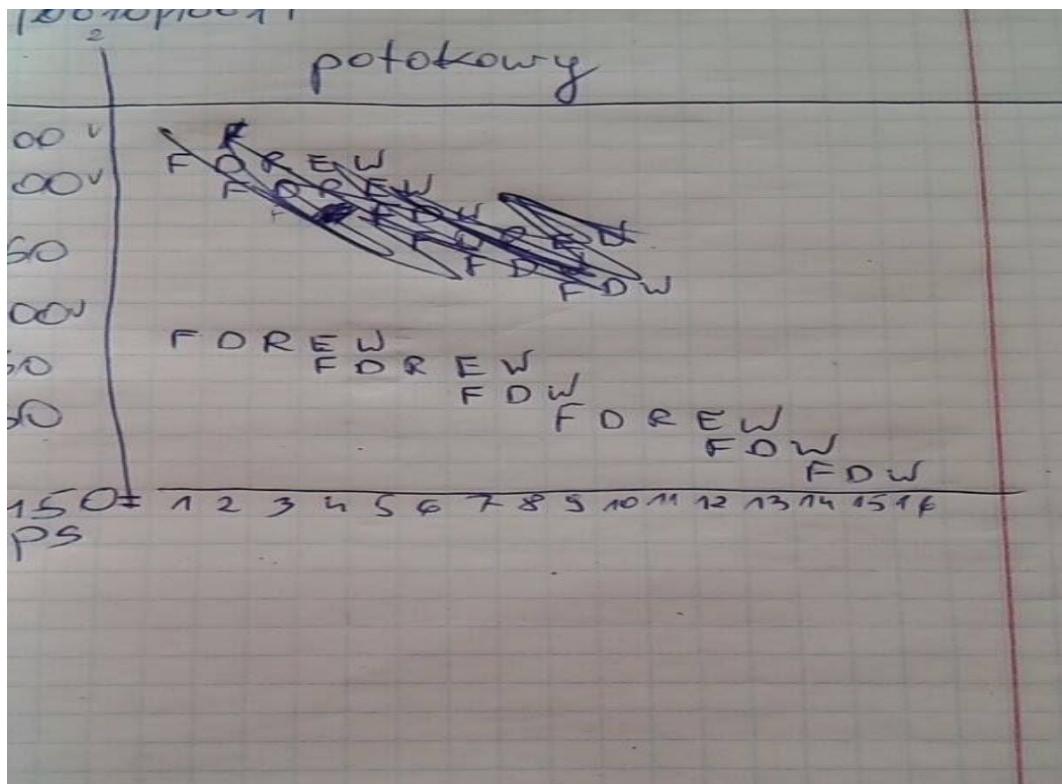
50	100	150	200	250	300	350	400	450	500	550	600	650	7/00	750	800	850
----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------	-----	-----	-----

Mikrooperacje:

F	D	R	E	W												
			<b>F</b>	<b>D</b>	R	<b>E</b>	<b>W</b>									
						<b>F</b>	<b>D</b>	<b>W</b>								
									<b>F</b>	D	R	<b>E</b>	<b>W</b>			
										<b>F</b>	<b>D</b>	R	<b>E</b>	<b>W</b>		
													<b>F</b>	<b>D</b>	<b>W</b>	

Odp: 850 ps

A CZY TO POWYŻSZE NIE POWINNO BYĆ ZROBIONE PRZYPADKIEM TAK ? :



16 \* 50ps ? // To jest na innym indeksie // Byłem na konsultacjach, to na zdjęciu jest poprawnie  
 // Widziałem pracę na konsultacjach - było full punktów za takie rozwiązanie // Również potwierdzam -> inny index...

KOLEJNY PRYKŁAD:

Kota 1. indeks 226045

1.

$h_3$	6	0110	add	90r1	,	90r1
$h_1$	4	0100	add	90r1	,	90r0
F		1111	ld	\$1	,	90r3
E		1110	add	90r3	,	90r1
$h_0$	5	0101	ld	\$0	,	90r1
$h_4$	2	0010	add	90r0	,	90r1

$$(i) (4 \cdot 5 + 2 \cdot 3) \cdot 50 \text{ [ps]} = 1300 \text{ [ps]}$$

(ii) FDREW



$$18 \cdot 50 \text{ [ps]} = 900 \text{ [ps]}$$



## Zadanie 2. 2016 termin 0

2. (5p) Jest dana liczba binarna  $(1+bbbb \cdot 2^{-23} + 3 \cdot 2^{-23}) \cdot 2^{-d}$  gdzie  $bbbb$  to 4 najmłodsze bity, zaś  $d$  to najmłodsza cyfra numeru indeksu. Zapisać tę liczbę w postaci zgodnej z normą IEEE 754, a następnie obliczyć sześcian tej liczby korzystając z przybliżenia  $(1+x)^3 \approx 1+3x$  dla  $x \ll 1$ .

// Według mnie to zadanie powinno zostać zrobione tak:  
// #Konsultacje: Byłem na konsultacjach, mam za to zadanie max  
// Po pierwsze warto zauważać że oznaczenia: **h-hex**, **d-dec**, **b-bin**  
// Po drugie nasz **X=Z\*M\*2^E**, co bardzo ułatwia całe zadanie  
// Weźmy przykładowy indeks **212757**  
**h0=d=0x7=7**  
**bbbb=0x7 =7 = 0111b // czemu bierzemy tylko ostatnią cyfrę, a nie 4 ostatnie bity całego indeksu ?**  
)  
    \* 2<sup>(-d)</sup> // założenie  
**X=(1 + 0111b\*2<sup>(-23)</sup> + 3\*2<sup>(-23)</sup>) \* 2<sup>(-7)</sup> // to bbbb to trochę zmyła, jest to po prostu 7**  
**X=(1 + 7\*2<sup>(-23)</sup> + 3\*2<sup>(-23)</sup>)=(1 + bbbb\*2<sup>(-23)</sup> + 3\*2<sup>(-23)</sup>) \* 2<sup>(-7)</sup>**  
**X=(1 + 10\*2<sup>(-23)</sup>) \* 2<sup>(-7)</sup> // liczba ta jest postaci 1,0000....1010 \* 2<sup>(-7)</sup>**

// jakby ktoś się zastanawiał czemu z **10\*2<sup>(-23)</sup>** sie zrobiło 00...1010 zamiast 00...10 to to **0** z **10\*2<sup>(-23)</sup>** jest zapisane dziesiętnie..... :D

// @UP, przecież masz napisane wyżej oznaczenia, jeżeli na końcu nie ma 'b', to nie jest to \*reprezentacja binarna

// IEEE754 1/8/23

**BIAS = 2^(E-1)-1 = 2^7 -1 = 127 // BIAS - obciążenie CZEMU E = 8? ->linijka wyżej**

**E = -7 + BIAS = 120 = 01111000b**

**0 01111000 0000 0000 0000 0001 010 // X**

**Mantysa na jakiej podstawie została tu obliczona? No 10\*2<sup>(-23)</sup> Zapisz sobie 10 jako binarna 1010 w NB wiec nie trzeba wiodacego 0 i przesun przecinek o 23 w lewo xD**

// Teraz mamy zapisaną naszą wartość X w IEEE754, kolejno należy obliczyć jej sześcian  
**X^3=[(1 + 10\*2<sup>(-23)</sup>) \* 2<sup>(-7)</sup>]<sup>3</sup> // korzystamy z własności że  $(1+x)^3 = 1+3x$  dla  $x \ll 1$**   
**/że co? -> POLECENIE CZYTAJ!!!**

**X^3=(1 + 30\*2<sup>(-23)</sup>) \* 2<sup>(-21)</sup> // to jest nasz sześcian, teraz możemy zapisać go podobnie jak wyżej w IEEE754 (o ile w ogóle trzeba)**

**BIAS = 127**

**E = -21 + 127 = 106 = 01101010b**

**0 01101010 0000 0000 0000 0001 110 // X^3**

**//A jak GRS jest ustawiony? NP 100**

# Nigdy nie będzie ustawiony bo masz coś\*2<sup>(-23)</sup>, sprarametryzowana wartość wejściowa jest zapisana w postaci **X=Z\*M\*2^E**, czyli w standardzie IEEE754, jest wytłumaczzone to na samym początku...

# PS. W przypadku gdyby GRS miało jakiekolwiek znaczenie (inne zadania) robisz roundZero.  
##czy mógłby ktoś dokładnie rozpisać zaokrąglenie dla poszczególnych wartości bitów GRS?

// To zależy od rodzaju zaokrąglenia:

- Dla roundZero liczy się tylko bit G

//Racja, dzięki.

### Czemu tu się wykładniki różnią?

# gdzie? Dla X i X^3? Jasne że powinny być inne, nie wiem czy ktoś zmienił czy co

# UWAGA! Mieliśmy podnieść całą liczbę do sześciadanu, nie tylko mantysę :x

Tzn dla  $x^3$  i  $x$  sa takie same, a przecież powinny być inne noo :D.

#Są inne. Źl3 patrzysz

Ej, a pamięta kto sposób z nakładaniem takiej jakby maski na wszystkie bity oprócz najstarszego? W celu obliczenia wykładnika z obciążeniem

# W sensie że, bierzesz eksponenta i negujesz każdy bit oprócz najstarszego tak, o to Ci chodzi?

# Eksponenta 8 bitowa = 0000 0000b

# BIAS = 0111 1111b, to jest właśnie  $2^{(8-1)} - 1$ , czyli 10000000b - 1b

A czy przydkiem nie trzeba obliczyć najpierw tego wyrażenia w nawiasie, żeby to uzyc jako mantysę?

# Ale po co? To jest idealnie to czego nam potrzeba, żeby zapisać w IEEE754

zauważ że  $1+10 \cdot 2^{-23} = 1,0000\dots1010$  czyli dokładnie tak jak powinna wyglądać nasza mantysa, to -23 w wykładniku nie jest przypadkowe.

Aa, dobra kumam dzięki, po prostu zapisujemy to 10 czy 30 na pozycji do -potęgi

# Tak dokładnie, pomimo że zapis początkowo wygląda dość skomplikowanie to jest to po prostu zapisanie IEEE w reprezentacji dziesiętnej, więc co jak co ale Patron ułatwił to zadanie :x

No zgadzam się, ja chciałem początkowo to wszystko obliczać, a to znacznie ułatwia

/// str 3, umiałby ktoś rozjaśnić trochę sytuację?

<http://imgur.com/bFswpmK> //wydaje mi się, że ma to być tak

+Co myślicie? // k

// b0b1b2b3 to są 4 najmłodsze bity indeksu, było napisane. BITY, nie cyfry czyli h0(16)=b0b1b2b3(2) (h0 zapisane binarnie) WOW no tak przecież jest zrobione

$$2. \quad (1 + 6666 \cdot 2^{-23} + 3 \cdot 2^{-23}) \cdot 2^{-d}$$

$$6 \ 6 \ 6 \ 6 = 0101_2 \quad d = 5_{10}$$

$$(1 + 5 \cdot 2^{-23} + 3 \cdot 2^{-23}) \cdot 2^{-5} = (1 + 8 \cdot 2^{-23}) \cdot 2^{-5} = (1 + 1000 \cdot 2^{-23}) \cdot 2^{-5}$$

$$N = 2^{k-1} - 1 = 127 \quad E = -5_{10} + 127 = 122_{10}$$

$$\left[ (1 + 8 \cdot 2^{-23}) \cdot 2^{-5} \right]^3 = (1 + 8 \cdot 2^{-23})^3 \cdot 2^{-15} = (1 + 3 \cdot 8 \cdot 2^{-23}) \cdot 2^{-15} =$$

$$= (1 + 24 \cdot 2^{-23}) \cdot 2^{-15} = (1 + 11000_2 \cdot 2^{-23}) \cdot 2^{-15}$$

$$E = -15_{n_0} = -15 + 127 = 112 \text{ eV}$$

$$\begin{array}{r|l}
 1 & 1 \quad 2 \quad 0 \\
 5 & 6 \quad 0 \\
 2 & 8 \quad 0 \\
 \hline
 1 & 4 \quad 0 \\
 7 & 1 \\
 3 & 1 \\
 1 & 1 \\
 0 & \\
 \end{array}
 \quad
 \begin{array}{l}
 12 \\
 \hline
 B = 01110000
 \end{array}$$



### Zadanie 3. 2016 termin 0

3. (3p) Przedstawić architekturę jednostki wykonawczej procesora o organizacji  $h \% 3 = \dots : 0$  – akumulatorowej uogólnionej, 1 – uniwersalnej/LS, 2 – stosowej (% oznacza operator modulo).

<http://www.yoyoyo.w8w.pl/architekturaprocesora.html> Ktoś ma lepsze źródło?

**POLECAM KSIAZKE BIERNATA 32 strona wydanie 2002 moge wrzucic na grupę kierunku 23 oraz 24 slajd -> 03-ak2-wyklad3**

[http://imgur.com/a/P\\_nNvB](http://imgur.com/a/P_nNvB)

Czyli konkretnie to kazdy pierwszy rysunek ze slajdów? No jaha.

// co w ogóle oznaczają te wszystkie Rejestr/Memory, BM, miCR i tak dalej? :(

MB-Memory Buffer

CR- Condition Register

ALU - Arithmetic and Logical Unit, jednostka wykonawcza (np. FPU, IU)

Wystarczy narysowac ta jednostke czy przebieg tego jak w wykładzie?

//Jeżeli pokazałeś wszystkie elementy na jednym rysunku to wystarczyło (opis słowny dawał całe NULL pkt) 0 bez rysunku???

//Nie do końca - w architekturze stosowej napisałem o rozkazach push/pop oraz wskaźniku esp i miałem 2pkt

**akumulatorowa AC wg Janusza - dla mnie wygląda jak akumulatorowa rozszerzona wg Patronika -> ksiazka JAnusza str 32 rok 2002 ??? ktos moze zweryfikowac, czy sie nie myle?**

– Jeden argument operacji musi byc umieszczony w wyroznionym rejestrze procka, w którym jest tez zapamietany wynik operacji. Poniewaz mozna w nim podwojnie wykorzystac ten rejestr nosi on nazwe akumulatora. Drugi argument pobrany z pamieci idzie do bufora MBR, niedostepnym dla programu.

**Akumulatorowa uogólniona u Patronika - w ksiazce Biernata jako rejestr-pamiec**

Architektury rejestr-pamiec cechuje umieszczenie przynajmniej jednego argumentu w rejestrze procka. Drugi argument jest pobierany z innego rejestru procesora lub pamieci, tam tez zostaje umieszczony wynik działania. Mlejsce drugiego argumentu pelni funkcje uogólnionego akumulatora. -> ksiazka Janusza takiej tez uzywamy na labach **add \$1, %rax add \$1, jakisInt, add %rax, jakisInt**

Mam pytanie do rysunku z linka <http://imgur.com/a/PnNvB>

A mianowicie R to jest akumulator? Jesli tak to w jaki sposób jest w nim zapisany wynik, skoro nie ma do niego strzałki? >>>>W ktorej architekturze? Akumulator masz po lewej albo na dole

### **uniwersalna R+M**

– zarówno argumenty jak zrodlowe operacji jak tez wynik sa umieszczone w rejestrach procka lub w pamieci. Konieczne jest uzycie MBR, niezbednych podczas wykonania operacji z wiecej niz jednym argumentem pochadzaczym z pamieci.

Moze byc, ale nie musi byc akumulatora. MOze byc jeden rejestr jedna pamiec lub oba te same.

### **Rejestrowa L/S**

Jak w uniwersalnej tylko same rejesty ogólnego przeznaczenia. Odstepstwem od reguły sa instrukcje komunikacji z pamiecia load/store (chyba chodzi o np. mov).

### **Stosowa**

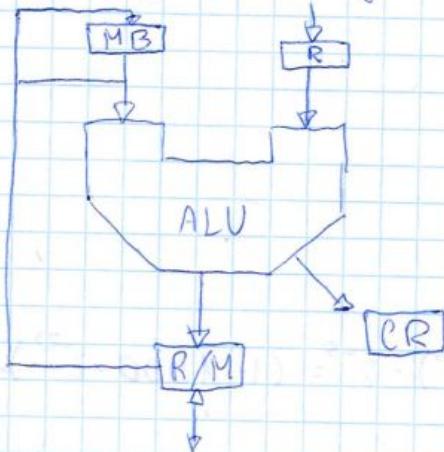
Argumenty operacji wykonywanych przez maszyne stosowa (stack machine) sa umieszczone w bloku pamieci lub pliku rejestrow skonfigurowanym jako bufor LIFO (Last in First out) zwany stosem obliczeniowym. Dostepny jest argument ze szczytu, wiec kolejny moze byc uzyty po pobraniu poprzedniego. Rozkaz niszczy dane których uzywał. Wykorzystanie FPU, funkcje - taki model jest symulowany w maszynie wirtualnej.

// #Konsultacje: w zadaniu należało narysować rysunek, za opis słowny dostawało się 0 (czasami 1, zależy jak mu się chciało :x) -> a na PTM odwrotnie za rysunek 0 opisać trzeba xD

3.

architektura jednostki wątkowej:

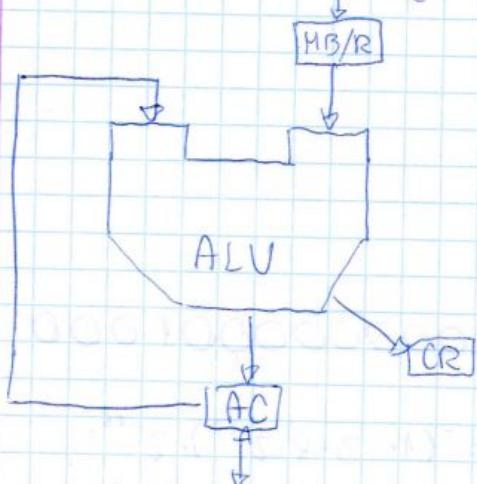
- akumulatorowej uogólnionej



Pierwszy argument zapisuje się w rejestrze procesora. Drugi argument albo jest w rejestrze albo w pamięci. Wymieniony jest w miejscu drugiego argumentu. Miejsce to pełni funkcję uogólnionego akumulatora.

Wynik zapisany jest w miejscu drugiego argumentu. Miejsce to pełni funkcję uogólnionego akumulatora.

- akumulatorowej i rozszerzonej



Jeden z argumentów zapisuje się w akumulatorze. Tam też zapisuje się wynik.

## Zadanie 4. 2016 termin 0

4. (5p) Jest dany fragment kodu pewnego procesu w systemie z segmentacją stronicowaną, z modelem programowym x86. Niech rozkaz rand (będący rozszerzeniem modelu) ładuje rejestr wartością zmiennej losowej wg rozkładu jednostajnego z przedziału od 0 do wartości z rejestru. (i) Ile takich procesów można równocześnie uaktywnić bez wystąpienia efektu migotania, jeżeli rozmiar dostępnej pamięci głównej wynosi 512MB?

```
mov $(h0*15+2), %eax
mov $0x10000, %ebx
mul %ebx
mov %eax, %ebx
mov $-1, %ecx

begin:
    mov %ebx, %eax
    rand %eax
    mov %eax, (%eax)
loop begin
```

Indeks : 218246

```
mov $(h0 * 15 + 2), %eax           // eax = 92
mov $0x10000, %ebx                  // ebx = 216 = 64kB | 210 = 1kB 220 = 1MB
                                         PAMIEC ADRESOWANA BAJTOWO (byte addressed memory)
mul %ebx                           // eax = 65536 * 92 = 64kB * 92 = 5888kB = 5,75MB
mov %eax, %ebx                     // ebx = 65536 * 92
mov $-1, %ecx                      // ecx = -1

begin:                                // Początek pętli
    mov %ebx, %eax                 // eax = ebx = 65536 * 92
    rand %eax                     // eax = losowanie z przedziału <0; 65536 * 92>
    mov %eax, (%eax)              // (eax) = eax
loop begin                            // dekrementacja ecx i skok jeśli ecx != 0
```

LINK: [https://en.wikibooks.org/wiki/X86\\_Assembly/Control\\_Flow#Loop\\_Instructions](https://en.wikibooks.org/wiki/X86_Assembly/Control_Flow#Loop_Instructions)

A dalej już przy dzieleniu: 512MB / 5,75MB = 89,043 procesów, więc 89 procesów.  
Gdybyście zostawili 64kb \* 92 zmieściłoby się dużo więcej procesów, a więc źle i nie współpracoby z komentarzem niżej.

Wg mnie losowanie je  
egzaminu podałeś rozwiązanie

Z tego co widzę to tutaj jest więcej zer, dlatego 2<sup>20</sup>. no ale są cztery zera, nie? XD

To mi sprawdził nie do tego egzaminu :PA co z h0\*15+2 ? Nie zauważył chyba, gdzie się podziało mnożenie 0x10000 przez to? :C

//czyli jak to obliczyć? Ktoś jeszcze na konsultacjach był i widział że to 512 jest wynikiem?

a CMP sprawdza czy wartosc nie jest zerem  
a skoro jest -1 to bedzie -2, -3, -4 ... - $2^{32}$  (max ujemna wartosc) i az przekräci się do 0  
(ostatecznie w takim wypadku coś to zmienia? )

Po prostu petla wykona sie  $2^{32}-1$  razy, tak jak pisal patronus w opisie drugiego terminu z  
zeszlego roku.

Liczba procesów:

512 MB =  $2^{29}$  [B] //dlaczego tutaj są bajty, a nie bity? // MB =  $2^{20}$ B, 512 =  $2^9$ ; mnożysz i tyle

#### **BYTE ADDRESSED MEMORY - pamiec adresowana bajtowo**

$2^{29} / (2^{16} * 92) = 2^{13} / 92 = \text{floor}(8192 / 92) = 89$  procesów

//I to jest rozwiązywanie żeby nie było migotania ? Czy nie trzeba robić odstępów między blokami? //  
a po co robić odstępy między blokami?

//Tak, bez migotania. Z migotaniem mogłoby być uruchomionych więcej procesów np .90. Choć  
//to też chyba zależy.  $8192/92=89.04347826$ . Bez migotania zaokrąglamy w dół.

//a gdzie w wyniku jest uwzględnione te  $2^{32}-1$  obrotów pętli?-> //to ile razy wykona sie petla  
nie ma wpływu na rozmiar zbioru roboczego procesu

Na chłopski rozum:

Efekt migotania = efekt szamotania - Suma zbiorów roboczych > rozmiar dostępnej pamięci

// migotanie to nie to samo co szamotanie? ->

[http://janek.ae.krakow.pl/~wiluszt/SO/sop\\_pamiec.pdf](http://janek.ae.krakow.pl/~wiluszt/SO/sop_pamiec.pdf)

Zbiory robocze to zapotrzebowanie procesu na pamięć w stanie wykonywania - co to może być w  
tym zadaniu? Żadnej danej nie ma podanej. //no ta funkcja rand... ona sypie z przedziału... Czyli  
to jest ( $2^{16} * 92$ ) ?? // wwwydaże mi sie ze tak...

#### **INSTRUKCJE UŻYWANE W KODZIE:**

[http://www.cs.put.poznan.pl/adanilecki/inline\\_asm/doda.php#inne-spis](http://www.cs.put.poznan.pl/adanilecki/inline_asm/doda.php#inne-spis)

[https://pl.wikibooks.org/wiki/Asembler\\_x86/Instrukcje/Arytmetyczne](https://pl.wikibooks.org/wiki/Asembler_x86/Instrukcje/Arytmetyczne)

Poczekajcie mam pytanie... Patrząc na opis tego co było rok temu - on jeszcze dał zadanie na  
drzewo CSA... ktoś pamięta wzory na policzenie opóźnienia tego drzewa? I ile FA trzeba uzyc?

// zaraz to znajdę w materiałach Janusha

// Iguano, tutaj jest częściowa odpowiedź:

[http://lucc.pl/inf/architektura\\_komputerow\\_1/2004\\_-\\_szybkie\\_sumatory\\_dwuargumentowe.pdf](http://lucc.pl/inf/architektura_komputerow_1/2004_-_szybkie_sumatory_dwuargumentowe.pdf)

// na stronie 12, charakterystyka AT (może coś pomoże xD)

(10)

$$h_0 \cdot 15 + 2 = 5 \cdot 15 + 2 = 77_{10}$$

$$10000_{16} = 16^4 = 2^{16}$$

rand losuje z:  $[0, 77 \cdot 2^{16}]$

$$\text{prawie} = 512[\text{MB}] = 2^9[\text{MB}] = 2^9 \cdot 2^{20} [\text{B}] = 2^{29} [\text{B}]$$

$$\text{kilkad procesów} = \left\lfloor \frac{2^{29}}{77 \cdot 2^{20}} \right\rfloor = \left\lfloor \frac{2^9}{77} \right\rfloor = \left\lfloor \frac{512}{77} \right\rfloor = 106$$

## Zadanie 5. 2016 termin 0

5. (4p) Wymienić z przykładami tryby adresowania architektury x86

LINK:

<https://drive.google.com/folderview?id=0Bxrgd6x3m9NiZUIxQ1JORmhWkE&usp=sharing&tid=0Bxrgd6x3m9NiUkZQXy1aSDI5YUU> (wykład 7, strona 26) // wklejone poniżej

1. Natychmiastowe: `mov $a, %eax`
2. Bezpośrednie: `mov a, %eax`
3. Pośrednie: `mov (%ebx), %eax`
4. Pośrednie z przemieszczeniem: `mov a(%ebx), %eax`
5. Pośrednie indeksowe: `mov (%ebx,%edi), %eax`
6. Pośrednie indeksowe z przesunięciem:  
`mov a(%ebx,%edi), %eax`
7. Pośrednie indeksowe z przemieszczeniem i skalowaniem:  
`mov a(%ebx,%edi,2), %eax`

Z konsultacji: Wykuć nazewnictwo z tego slajdu po wyżej - nie kombinować z bazowymi górnymi, tak ma być i koniec. ADDR to eax w każdym przypadku. Na pytanie co to jest addr odpowiedział, że **wartość arytmetyczna?** Nie do końca wiem o co mu chodziło, dodał jednak że nie zwracał na to uwagi i jeśli ktoś dobrze nazwał adresowania tak jak tutaj to był full. PS Tabelka z tamtego roku jest chuj warta //a jak się używało tej tabelki to bardzo obcinał punkty? // za nawet jeden błąd w których adresowaniu dawał 1 pkt za całe zadanie

// Opisy na slajdach wyżej (w prezentacji) - nazwy adresowań nie pokrywają się z tymi z tego slajdu tak nawiąsem, ale gdy będzie trzeba wypisać tak jak to było na terminie zerowym to też polecam przepisać to co jest powyżej (te 7 punktów) bez względu na wszystko.

// Ja pisałem jako **ADDR adresy źródłowe** i też dał maksa więc skoro ktoś miał inaczej również z maksymalną ilością punktów to faktycznie na to nie zwraca uwagi

**Lokalność czasowa** – tendencja do powtarzania odwołań, realizowanych w niedawnej przeszłości (realizacja pętli, referencje do tablicy).

**Lokalność przestrzenna** – tendencja do odwołań do obiektów w obszarze adresowym obejmującym obiekty wcześniej użyte w programie ( kolejne rozkazy programu, elementy regularnej struktury danych, tablice translacji adresów obszar roboczy stosu programowego).

# Konsultacje: aby dostać max punktów za to zadanie należało dokładnie opisać jak nazywa się adresowanie, np:

`mov a(%ebx, %edi, 2), %eax` - bazowo-indeksowe z przemieszczeniem i skalowaniem

// Jeżeli gdziekolwiek jest coś w () to jest to adresowanie pośrednie, w innym przypadku bezpośrednie

5.

(4)

mov \$a, %eax - natychmiastowe

mov a, %eax - bezpośredni

mov (%ebx), %eax - pośrednie

mov a(%ebx), %eax - pośrednie z przemieszczeniem

mov (%ebx, %edi), %eax - pośrednie indeksowe

mov a(%ebx, %edi), %eax - pośrednie indeksowe z przesunięciem

mov a(%ebx, %edi, 2), %eax - pośrednie indeksowe z przemieszczeniem i skalowaniem

<b>mov _____</b>	<b>opis</b>	<b>ADDR=</b>
mov \$5, %eax	natychmiastowe	brak, w eax jest cyfra 5
mov a, %eax	bezpośrednie	a
mov (%ebx), %eax	pośrednie	%ebx
mov a(%ebx), %eax	pośrednie z przemieszczeniem	a + %ebx
mov (%ebx, %edi), %eax	pośrednie indeksowe	%ebx + %edi
mov a(%ebx, %edi), %eax	pośrednie indeksowe z przemieszczeniem	a + %ebx + %edi
mov a(%ebx, %edi, 2), %eax	pośrednie indeksowe z przemieszczeniem i skalowaniem	a + %ebx + 2 * %edi

+-

## **z ` Zadanie 6. 2016 termin 0**

**6. (4p) Przedstawić  $h_0 \bmod 2=0$ ; definicję efektu szamotania,  $h_0 \bmod 2=1$ : heurystykę jak mu zapobiec**

**Efekt szamotania-** Suma zbiorów roboczych procesów aktywnych > rozmiar dostępnej pamięci

**Heurystyka-** Nie wymieniaj bloku, który jest częścią zbioru roboczego aktywnego procesu i nie uaktywniaj procesu, którego zbiór roboczy nie może zostać w całości odwzorowany w pamięci głównej.

### **Szamotanie**

proces się szamoce jeśli spędza więcej czasu na stronicowaniu niż na wykonaniu.

*Wiki:*

Stan procesu, w którym spędza on więcej czasu na oczekiwanie na brakujące strony pamięci niż na faktycznym wykonywaniu obliczeń, co znaczenie spowalnia jego działanie.

*Wykład:*

**Suma zbiorów roboczych procesów aktywnych > rozmiar dostępnej pamięci.**

*Zapobieganie szamotaniu:*

- Szamotanie można ograniczyć przez zastosowanie algorytmu zastępowania lokalnego lub priorytetowego (nie likwiduje to jednak problemu).
- Szamotanie można zlikwidować, dostarczając procesowi tylu wolnych ramek ilu potrzebuje do wykonania określonego fragmentu programu (np. procedury, pętli).

## **Heurystyka**

*Wykład:*

Nie wymieniaj bloku, który jest częścią zbioru roboczego aktywnego procesu i nie uaktywniaj procesu, którego zbiór roboczy nie może zostać w całości odwzorowany w pamięci głównej.

#Moja definicja

W momencie kiedy jakiś obiekt jest żądany, istnieje duże prawdopodobieństwo że obiekty leżące obok niego(w pamięci) także będą żądane w najbliższym czasie.

// i co uznał Ci to ? #Moja definicja

// Przecież nie było nawet takie pytania w tym roku

Definicja ze Stallingsa :

Zgodnie z zasadą lokalności, odniesienia do pamięci mają tendencję do grupowania się. W ciągu długiego czasu wykorzystywane ugrupowania zmieniają się, jednak w krótkim czasie procesor przede wszystkim pracuje z ustalonimi ugrupowaniami.

**DOBRA RADA:** Piszcie tylko definicje itd krótkie ale ZGODNE ze slajdami całkowicie. Najczęściej to daje max pkt nawet za napisanie jednej liniki. Nie wymyślajmy definicji `z chujwiekad ani tabelek z adresowaniami po swojemu (nawet jak one się teoretycznie zgadzają:)).

## 6. Efekt szemotowania

Proces się szemodzie, kiedy więcej czasu spędza na oczekiwaniu na bieżącące stony pomyęci niż na feletyzowaniu ukończenia obliczeń.

Dzieje się tak, gdy same zbiorów procesów elitywnych jest wiele od rozmiaru dostępnego pomyęci. Zapobiegamy temu, skoncentrujając procesy w tyle różnych ramach pomyęci, że potrafią do ukończenia konkretnego fragmentu programu.

## Heurystyka

Nie wgnijenie się bloku, który jest częścią zbioru robozegowego elitywnego procesu, ani nie dotyczy się procesu, którego zbiór robozry nie może zostać w pełni włączony do pomyęci głównego:

# Koło 2015, TERMIN 0

Organizacja i Architektura Komputerów. Egzamin, termin 0. 12.06.2015

Czas: 40 min. Używanie kalkulatorów: zabronione. Do notatek służy druga strona kartki. Przejrzysty zapis obliczeń ułatwia mi rozstrzyganie orzynadków niejednoznacznych. Życzę Wam powodzenia – Piotr Patronik

Imię i nazwisko: ..... Numer indeksu: ..... h<sub>3</sub>h<sub>2</sub>h<sub>1</sub>h<sub>0</sub> Punkty: ...../25

1. (5p) Jest dany procesor o 4-bitowym słowie rozkazowym i poniższym kodowaniem rozkazów. Zapisać program w postaci mnemoników i podać wartości w rejestrach po wykonaniu 4 rozkazów (zapisanych szesnastkowo): h<sub>1</sub>, h<sub>2</sub>, h<sub>1</sub>, h<sub>0</sub>, zakładając, że wartości początkowe rejestrów wynosiły 0.

i	i	v	0	ld	\$v,	%ri	ti = v
i	i	j	1	add	%ri,	%rj	tj = ri+rj

2. (4p) Narysować schemat sumatora prefiksowego (h<sub>0</sub>%3)+3-bitowego w architekturze h<sub>0</sub> mod 3=0 – Kogge-Stone'a, h<sub>0</sub> mod 3=1 – Sklansky'ego, h<sub>0</sub> mod 3=2 – Han-Carlson'a i przedstawić jego działanie dla dwóch dowolnie wybranych (różnych i niezerowych) wektorów wejściowych.

3. (6p) Są dane dwie liczby zmiennoprzecinkowe w standardzie podobnym do IEEE 754 z tą różnicą, że mantysa ma 7 bitów które są szesnastkowo zapisane jako A=h<sub>3</sub>h<sub>2</sub>h<sub>1</sub>h<sub>0</sub> i B=h<sub>1</sub>h<sub>2</sub>h<sub>3</sub>h<sub>4</sub>. Obliczyć i podać ich sumy i ilorazy, tj A+B i A/B, wyniki podać dziesiętnie i szesnastkowo w formacie źródłowym.

4. (5p) Jest dany fragment kodu pewnego procesu. Niech rozkaz rand ładuje rejestr wartością zmiennej losowej wg rozkładu jednostajnego z przedziału od 0 do wartości z rejestru. Ile takich procesów można równocześnie uruchomić, jeżeli rozmiar dostępnej pamięci głównej wynosi 1024MB, zaś system operacyjny zużywa 10·h<sub>2</sub> MB? (rozmiar strony wynosi 2+h<sub>3</sub> kB, rozmiar segmentu procesu 1+h<sub>1</sub> GB)

```
mov $h0*16+3, %eax
mov $0x10000, %ebx
mul %ebx
add %edx, %eax
mov %eax, %ebx

begin:
    mov %ebx, %eax
    rand %eax
    mov %eax, (%eax)
jmp begin
```

5. (5p) Podać definicję lokalności: h<sub>0</sub> mod 2=0 – czasowej, h<sub>0</sub> mod 2=1 - przestrzennej.

# Koło 2015, TERMIN 0

## ROZWIĄZANIA:

### Zadanie 1. 2015 termin 0

1. (5p) Jest dany procesor o 4-bitowym słowie rozkazowym i poniższym kodowaniem rozkazów. Zapisać program w postaci mnemoników i podać wartości w rejestrach po wykonaniu 4 rozkazów (zapisanych szesnastkowo):  $h_3, h_2, h_1, h_0$ , zakładając, że wartości początkowe rejestrów wynosiły 0.

$ii\ v\ 0$  ld \$v, %ri       $ri = v$   
 $ii\ j\ 1$  add %ri, %rj     $rj = ri + rj$

Voto 2. indeks 226045

1.

$h_3$	6	0 110	ld	\$1	, %r1
$h_2$	0	0000	ld	\$0	, %r0
$h_1$	4	0100	ld	\$0	, %r1
$h_0$	5	0101	add	%r1	, %r0

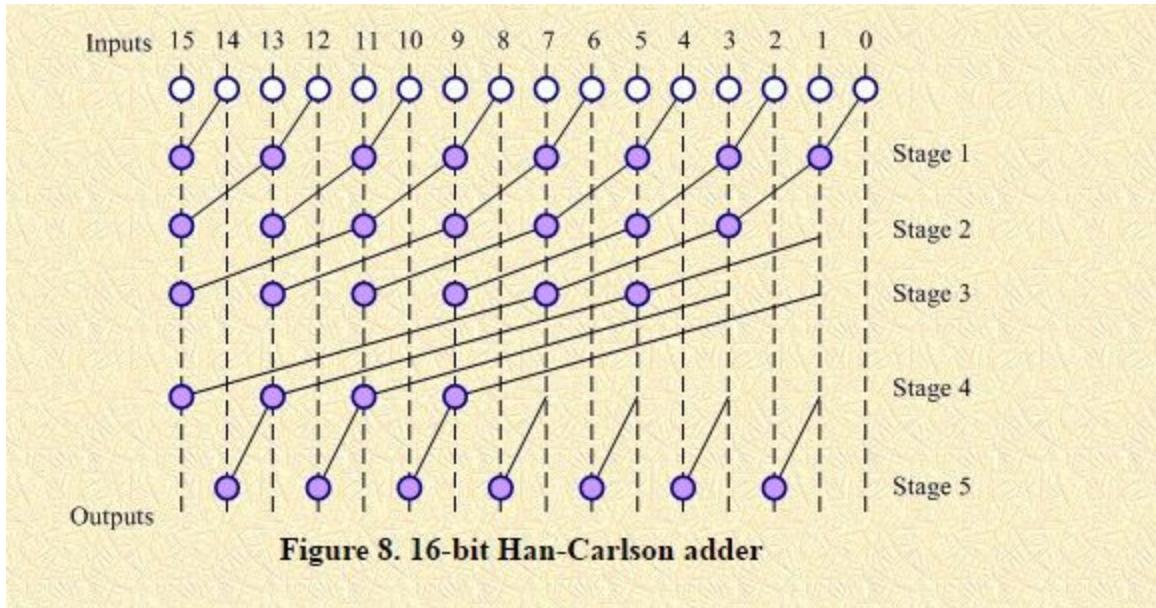
zwartosć rejestrów:  $+1=0$ ,  $r0=0$

ii to 2 bity, czyli można wybrać rejesty od  $r0$  do  $r3$ , czy ich wartości też nie powinnyśmy podać, w wyniku? Tj podać, że  $r2=0$  i  $r3=0$  //dla tego on napisał +1.. Co oznacza  $\geq 1$

## Zadanie 2. 2015 termin 0

2. (4p) Narysować schemat sumatora prefiksowego ( $h_0 \bmod 3=0$  – Kogge-Stone'a,  $h_0 \bmod 3=1$  – Sklansky'ego,  $h_0 \bmod 3=2$  – Han-Carlson'a i przedstawić jego działanie dla dwóch dowolnie wybranych (różnych i niezerowych) wektorów wejściowych.

gdyby ktoś szukał sumatora Han -Carlson'a



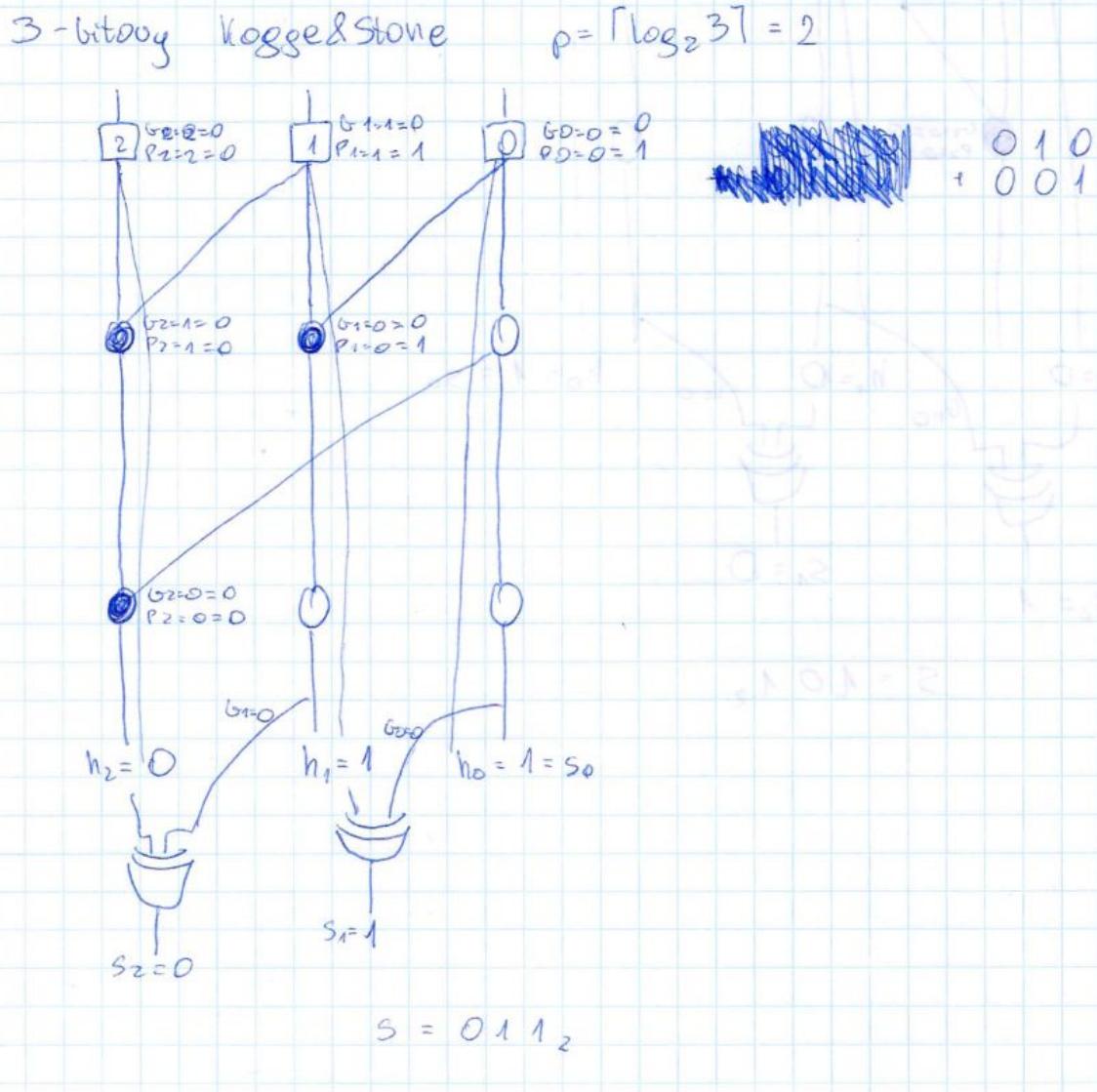
### GRAFY

<http://www.aoki.ecei.tohoku.ac.jp/arith/mg/algorith.html>

<https://www.slideshare.net/peeyushpashine/parallel-prefix-adders-presentation>

## Do zadania 2 - 2017

[http://inf.lucc.pl/architektura\\_komputerow\\_1/2009\\_-\\_7\\_szybkie\\_1101sumatory.pdf](http://inf.lucc.pl/architektura_komputerow_1/2009_-_7_szybkie_1101sumatory.pdf)

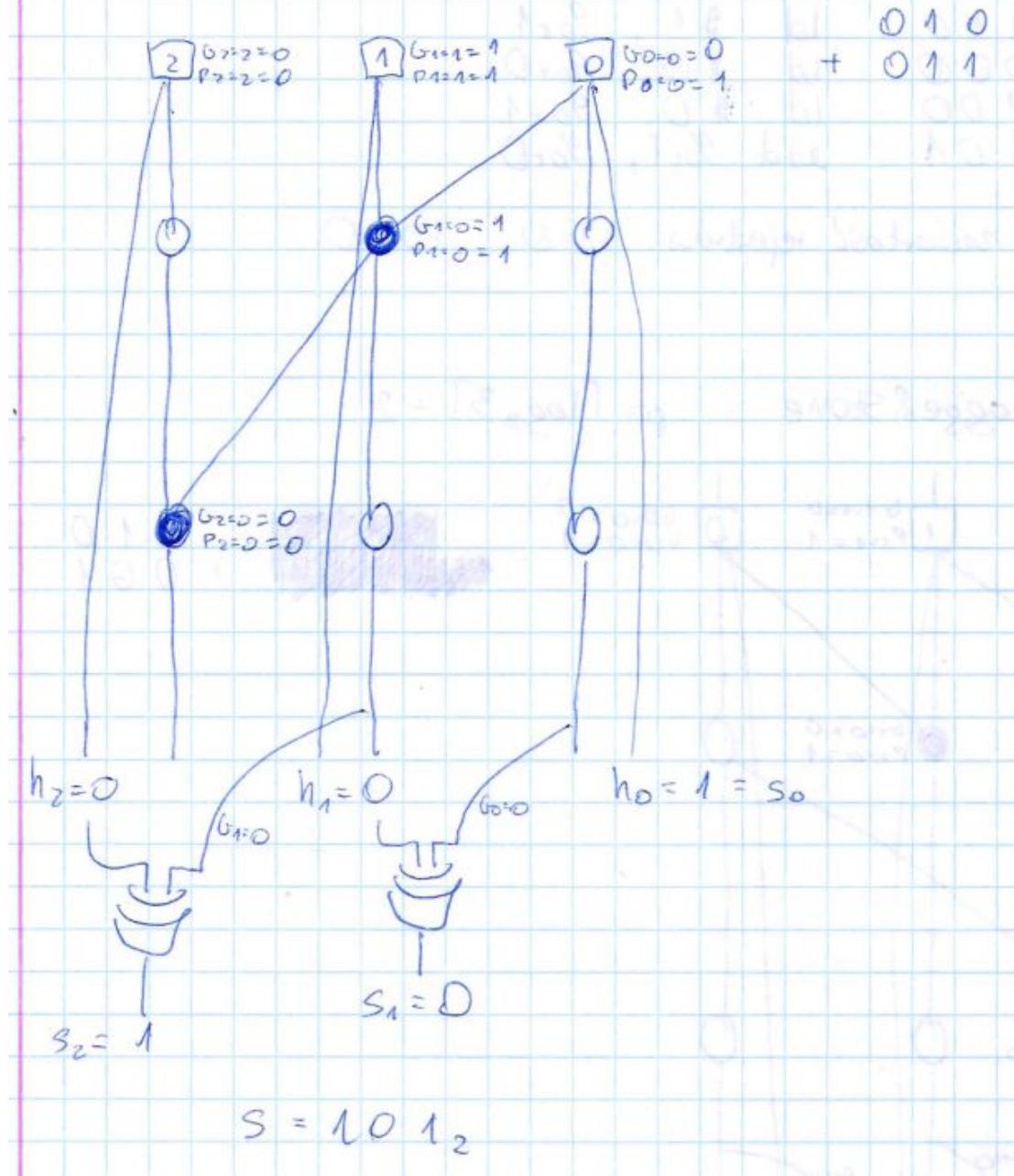


Może ktoś łopatologicznie wytłumaczyć jak liczyć te wartości G i P?

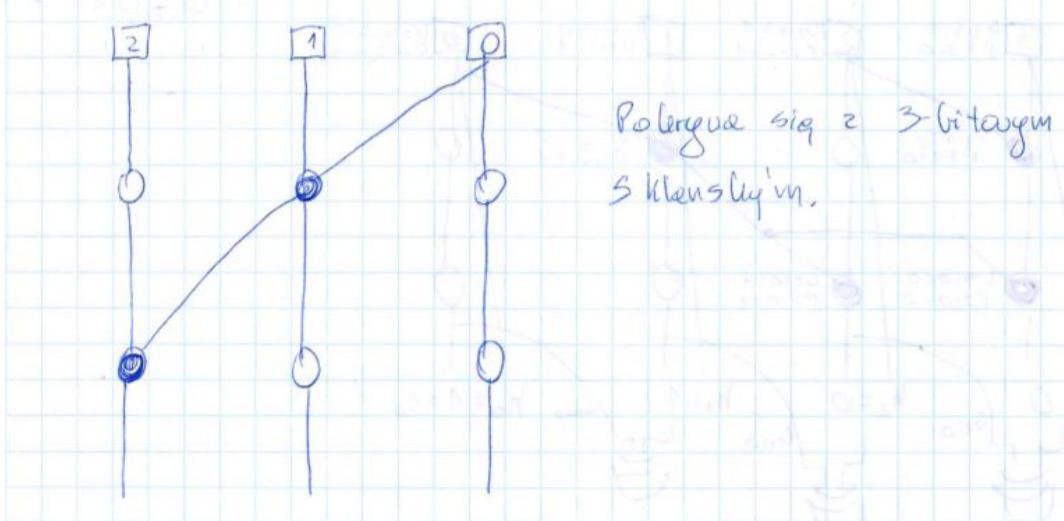
Ktoś może powiedzieć co oznacza G i P na początku drzewa bo P to chyba wartość mod a G?

3-bitowy Składowy

$$p = \lceil \lg_2 3 \rceil = 2$$



3-bitowy Han - Carlson



//linia ze środkowej kropki nie powinna iść na bit nr 3 (a nie nr 2)?

ŽLE - powinna być jedna kropka

IMO 2 kropki ale miedz tymi rzędami gdzie sa kropki to jeszcze pusty rząd jeden 3 puste kropki!

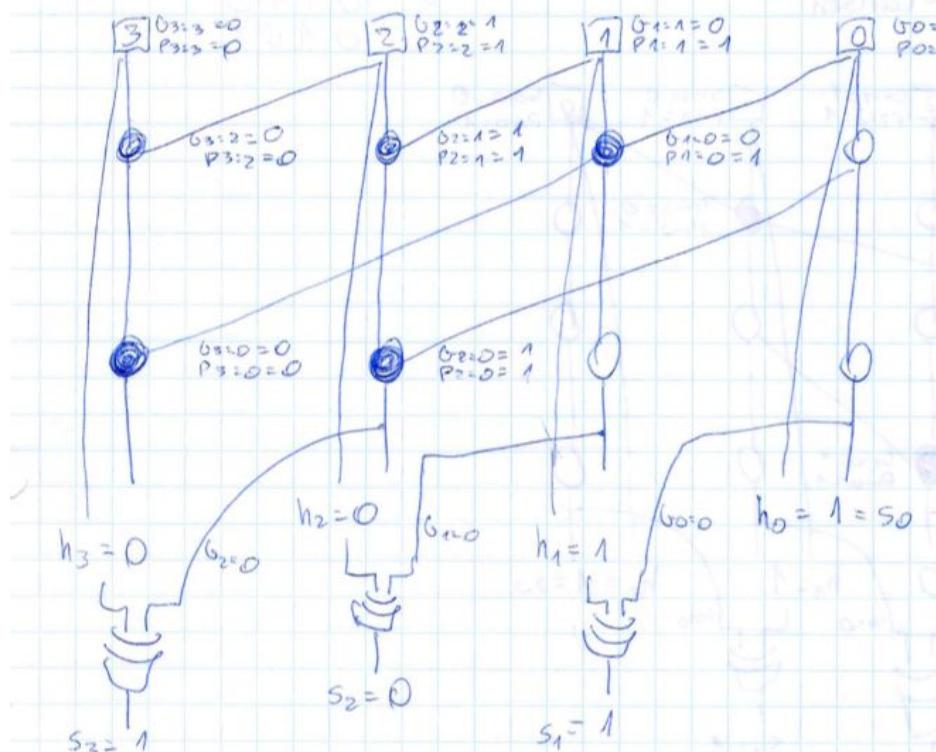
DOBRZE JEST

Ostatni rząd w han carlsonie zawsze jest taki sam

4-bitacyclic Kogge & Stone

$$p = \lceil \log_2 47 \rceil = 2$$

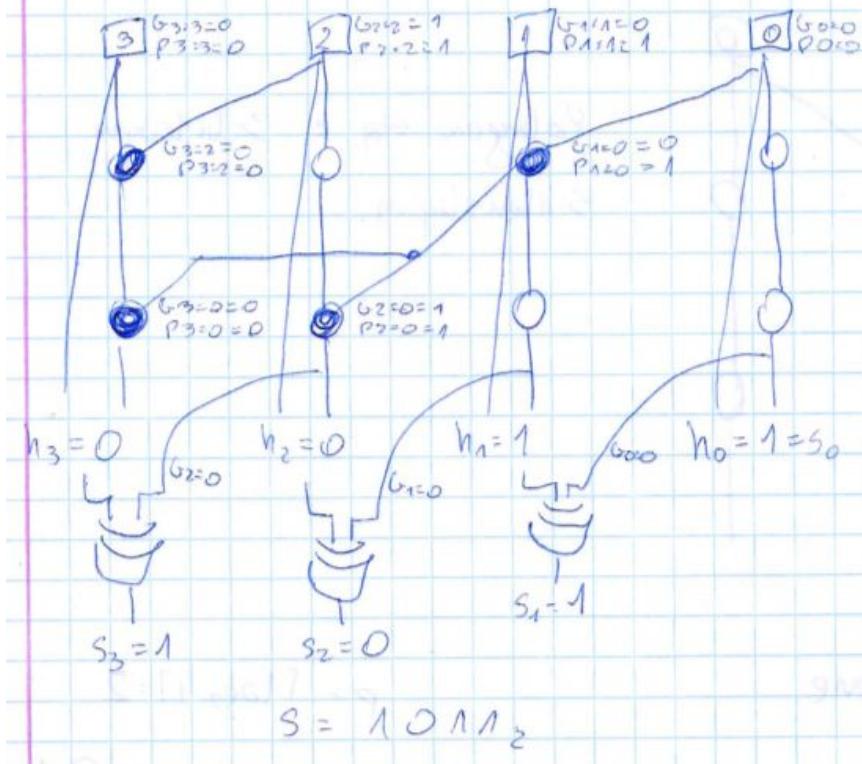
$$\begin{array}{r} 0110 \\ + 0101 \\ \hline \end{array}$$



4-Bit Gray Sleichung

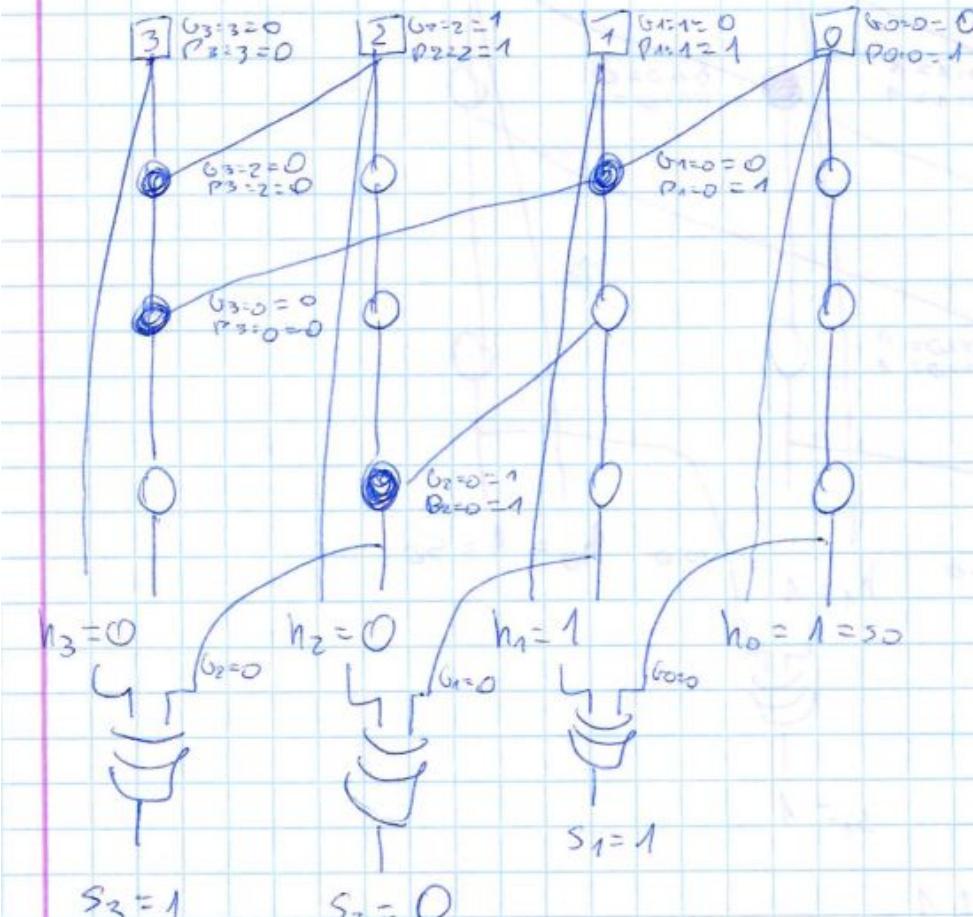
$$p = \lceil \log_2 4 \rceil = 2$$

0 1 1 0  
+ 0 1 0 1



4-bitong Hæn-Carlson

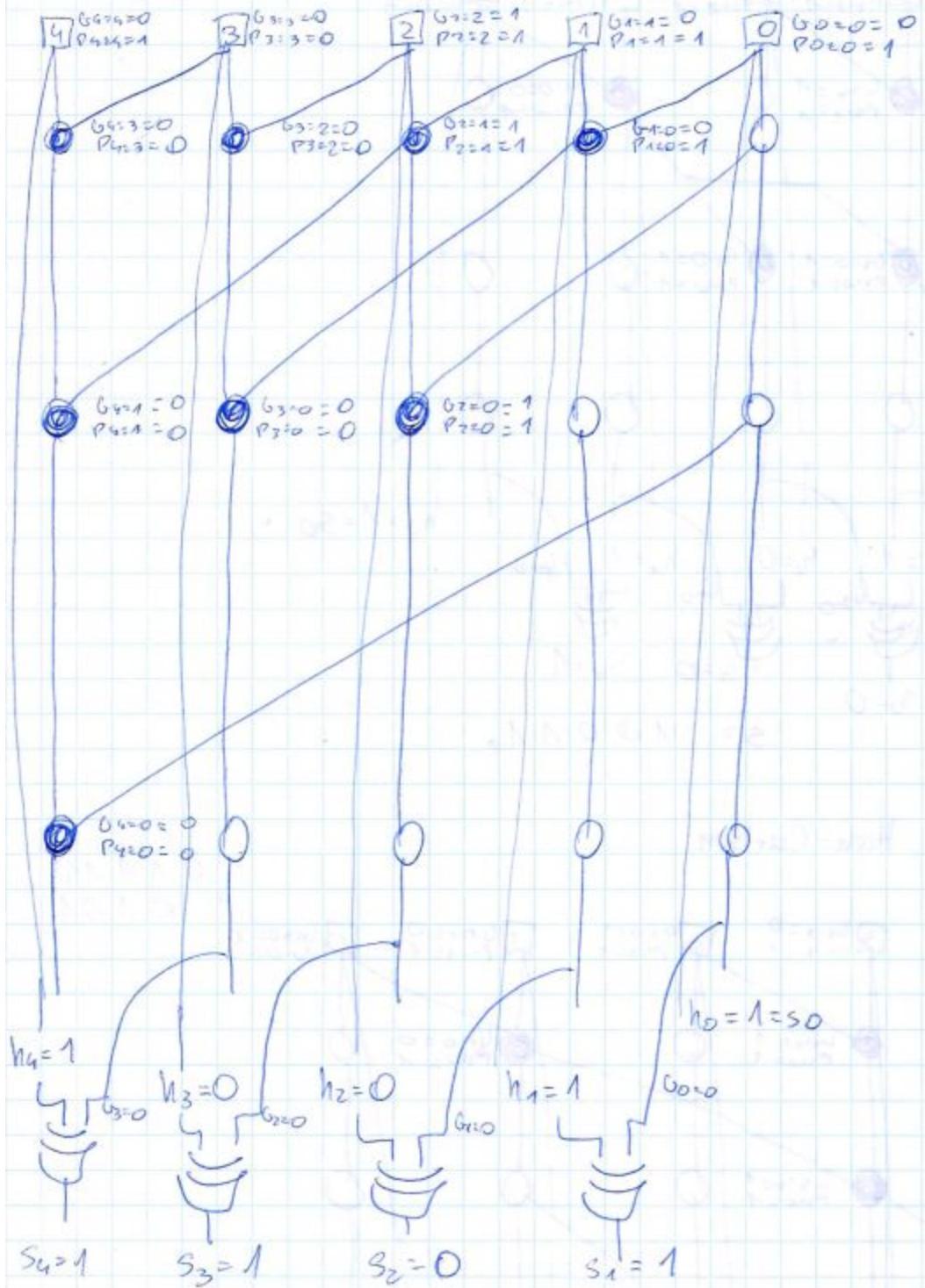
$$+ \begin{array}{r} 0110 \\ 0101 \end{array}$$



$$S = 1011_2$$

5 - vitawy KoggeStone  $p = \text{Flag}_2[5] = 3$

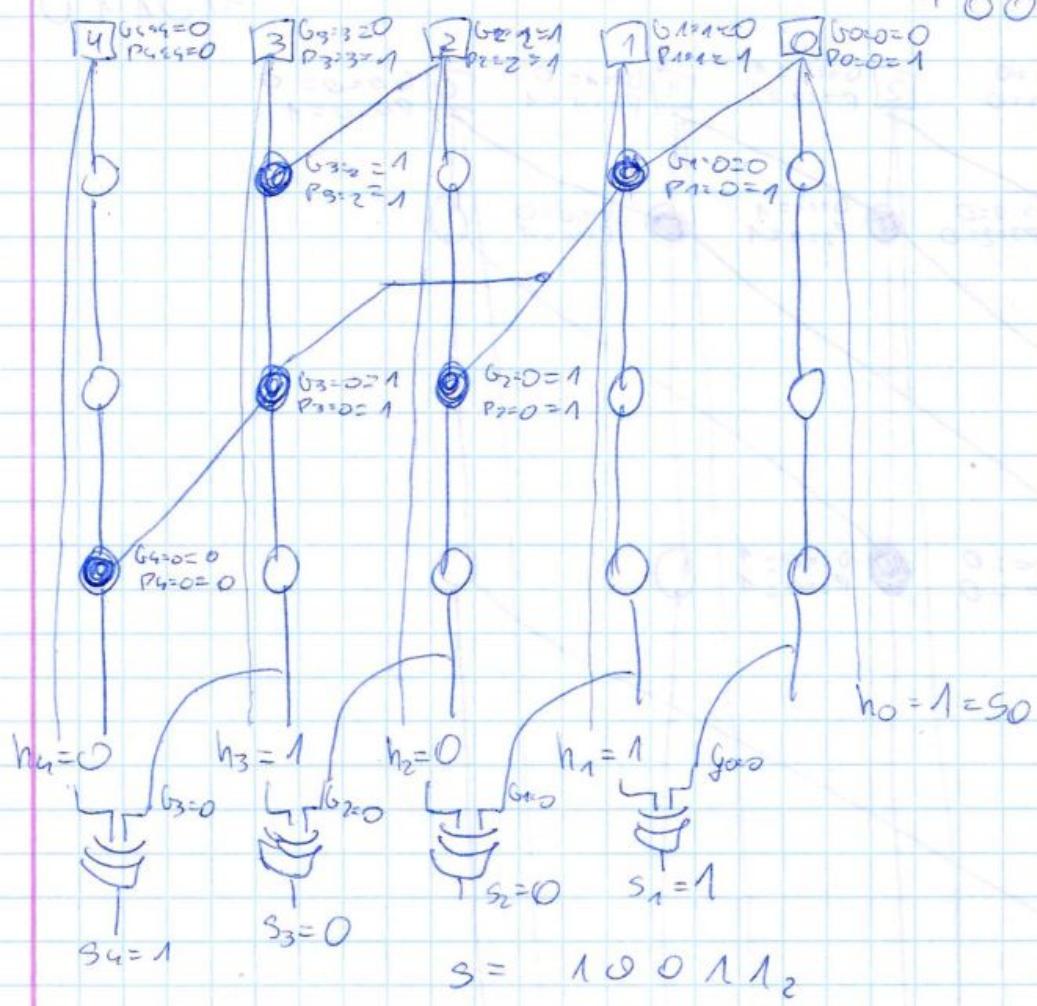
$\begin{array}{r} 00101 \\ -110110 \end{array}$



$$S = 11011_2$$

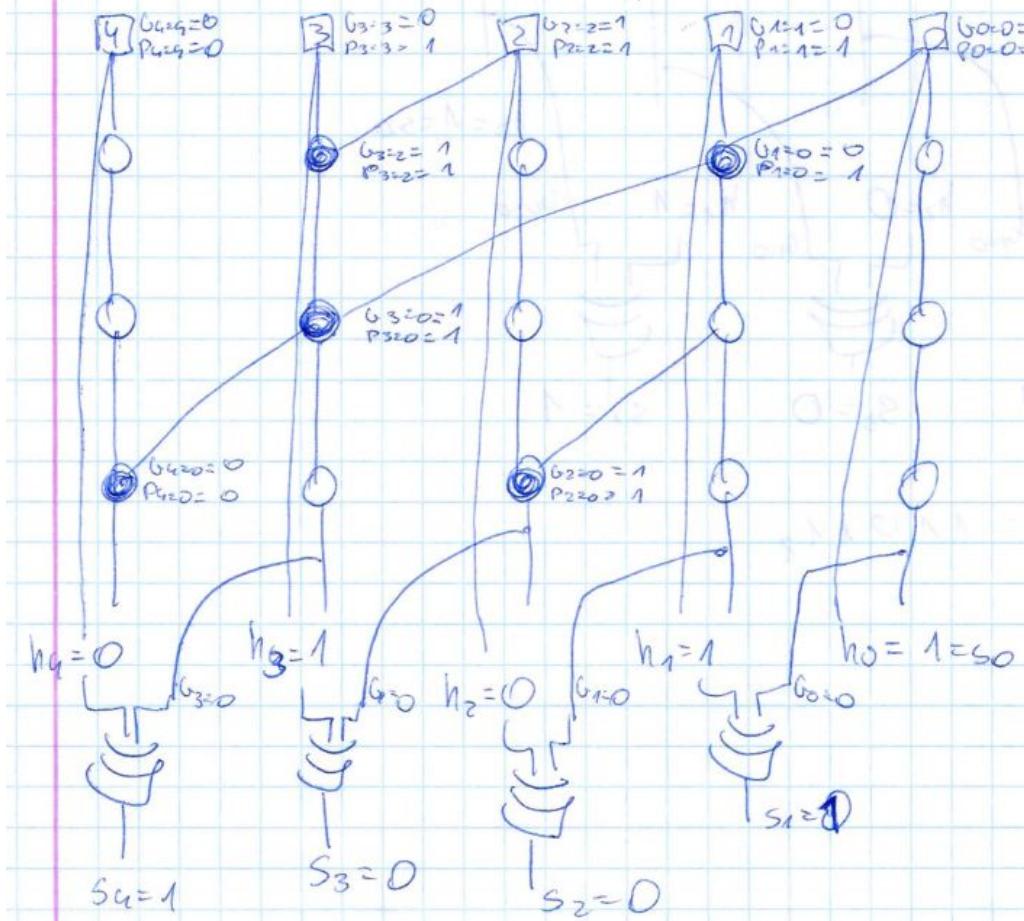
5-bitage Skilenskyg  $p = \lceil \log_2 5 \rceil = 3$

$$\begin{array}{r} 01110 \\ + 00101 \\ \hline \end{array}$$



5-Bit Gray Han-Carlson

$01110$   
 $-10101$



$$S = 10011_2$$

### Zadanie 3. 2015 termin 0

3. (6p) Są dane dwie liczby zmiennoprzecinkowe w standardzie podobnym do IEEE 754 z tą różnicą, że mantysa ma 7 bitów które są szesnastkowo zapisane jako  $A = h_3 h_2 h_1 h_0$  i  $B = h_3 h_2 h_1 h_0$ . Obliczyć i podać ich sumy i ilorazy, tj  $A+B$  i  $A/B$ , wyniki podać dziesiętnie i szesnastkowo w formacie źródłowym.

/Dużo osób o to pyta, więc napiszę tutaj. Z tego co zrozumiałem: mamy dwie liczby zmiennoprzecinkowe w standardzie podobnym do IEEE 754, więc one w tym standardzie już są - nie mamy ich na niego konwertować. Te liczby są zapisane w postaci 4 cyfr 16tkowych - pasuje, bo nasz standard ma 16 bitów (znak, 8 wykładnik, 7 mantysa), a każda cyfra 16tkowa ma 4 bity. Tak więc traktuję bity tych cyfr jako bity liczby IEEE 754 o mantysie długości 7.

1

Wykładnik jest w kodzie z obciążeniem  $+N$ , nie powinno się w nim wykonywać obliczeń takich jak suma dwóch liczb w  $+N$ , mnożenie itp, bo zwielokratniamy obciążenie. Przejście wykonujemy negując wszystkie bity oprócz najstarszego. +Możemy wykonywać w zaprzeczonym U2 obliczenia nie robiąc operacji NOT, ponieważ wracając do  $+N$  zaprzeczenie dokona się samo.

$$3. A = h_3 h_2 h_1 h_0 = 6045_{16}$$

$$226045$$

$$B = h_3 h_2 h_1 h_0 = 4062_{16}$$

$$A = 011000000|1000101$$

$$B = 010000000|1100010$$

(1)

$$A+B = 2^{E_A} (M_A + M_B \cdot 2^{E_B-E_A}) , E_A > E_B$$

$$E_A = 11000000_{-N} = 10111111_{\text{U2}}$$

$$E_B = 10000000_{-N} = 11111111_{\text{U2}}$$

$$E_B - E_A = \frac{11111111 - 10111111}{01000000_{\text{U2}}} = 00111111_{-N}$$

$$\begin{array}{r} 11111111 \\ - 10111111 \\ \hline 01000000_{\text{U2}} = -1 \cdot 2^6 = -64 \end{array}$$

$$M_B \cdot 2^{-64} \ll M_A \Rightarrow \text{pomijamy } M_B \cdot 2^{-64}$$

$$A \cdot B = 0|11000000|1000101$$

$2^7 + 2^6 - 127$

$$(1 + \frac{1}{2} - \frac{1}{32} + \frac{1}{128}) \cdot 2$$

6045<sub>16</sub>

$$A/B = \frac{M_A}{M_B} \cdot 2^{E_A - E_B}$$

$$\begin{array}{r} 10111111 \\ - 11111111 \\ \hline 11000000 \end{array} \approx 10111111_{10}$$

$$\frac{M_A}{M_B} = \frac{1,1000101_2}{1,1100010_2} = \frac{011000101_2}{011100010_2}$$

$$\begin{array}{r}
 & & 100011110 \\
 & & \overline{011011110} \\
 \overline{01100011110} & -D & 011100010_2 \\
 1100011110 & & \\
 + 011100010 & +D & \\
 \hline
 00101010000 & & \\
 - 100011110 & -D & \\
 0011011100 & & \\
 - 100011110 & -D & \\
 11111110100 & & \\
 - 011100010 & -D & \\
 00110101100 & & \\
 - 100011110 & -D & \\
 011001010100 & & \\
 - 100011110 & -D & \\
 010001000100 & & \\
 + 100011110 & -D & \\
 000100010100 & & \\
 - 100011110 & -D & \\
 101100010 & & \\
 \end{array}$$

R   S=1  
 11 02^8  
 n

$$\frac{M_A}{M_B} = 011011110 = 1,10111110 \approx 1,101111_2$$

$$E = E_A - E_B - 1 = 101111111 - 1 = 101111110_{10}$$

A/B: 01011111 01011111

$$(1 + \frac{1}{2} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32} + \frac{1}{64} + \frac{1}{128}) \cdot 2^{2 \times 6 - 8 + 16 - 32 + 64 - 128} = 127$$

5F 5F<sub>16</sub>

## Zadanie 4. 2015 termin 0

4. (5p) Jest dany fragment kodu pewnego procesu. Niech rozkaz rand ładuje rejestr wartością zmiennej losowej wg rozkładu jednostajnego z przedziału od 0 do wartości z rejestru. Ile takich procesów można równocześnie uruchomić, jeżeli rozmiar dostępnej pamięci głównej wynosi 1024MB, zaś system operacyjny zużywa  $10 \cdot h_2$  MB? (rozmiar strony wynosi  $2 + h_1$  kB, rozmiar segmentu procesu  $1 + h_1$  GB)

```
mov $h2*16+3, %eax
mov $0x10000, %ebx
mul %ebx
add %edx, %eax
mov %eax, %ebx

begin:
    mov %ebx, %eax
    rand %eax
    mov %eax, (%eax)
jmp begin
```

# Czy system też trzeba postronicowac??????

Przykładowy indeks : 200794

**Rozmiar strony : 2kB**

**Zużycie pamięci (system)  $7 * 10 = 70$  MB**

**Rozmiar segmentu procesu: 10GB**

yt

mov \$4\*16+3, %eax | %eax = 67

Mov \$0x10000, %ebx | ebx =  $2^{16}$

Mul %ebx | eax =  $67 * 2^{16} = 67\text{ kB} * 2^6 = 67 * 64 = 4288\text{ kB}$  // rozmiar zbioru

Stron\_na\_zbiór = rozmiar zbioru/rozmiar strony =  $4288\text{ kB}/2\text{ kB} = 2144$  //hej misiaki, jak wychodzi z przecinkiem to zaokrąglą się w górę, czy w dół? NP. dla  $2240/3$  jest  $746,(6)$  - chyba w dół, right? //W GÓRĘ!!!!

(1024MB-system)/2kB //A nie  $10 * 1024\text{ MB} !!!$  Jest 10 GB a nie 1GB

$1024\text{ MB} - 70/2\text{ kb} = 954 * 2^{20} / 2 * 2^{10} = 954 * 1024 / 2 = 976896/2 = 488448$  // ile pamiec mieści

stron //tutaj pytanie dlaczego  $2 * 2^{10}$ , czy samo po lewej  $2^{20}$  nie załatwia sprawy?

Nie bo dzięki temu masz wszystko w Bajtach

488448/2144 ~ 227 procesów

Ok, fajnie jakby ktoś to ładnie opisał teraz i uporządkował

//ej co się dzieje z ROZMIAREM segmentu procesu 10gb? Bo wyżej go nie uzyli w rownaniu?????

Jak mamy dzielenie mantys dajmy na to 1010 mantysa 1 i 0101 mantysa druga, czyli

1,1010:1,0101 i dzielimy, ale to w rzeczywistości są obie liczby dodatnie nie? Czyli jakby (0)1,1010:(0)1,0101? Bo już mi się zapomniało trochę z ak1

TAK

A mogliby ktos napisac tak na "chlopski rozum" j ak sie skaluje wykładniki przy dodawaniu?

Np jak mamy

1 0101 101

1 1010 110

Ja tam sobie to rozpisuje. Kod z obciążeniem tutaj  $2^3 - 1$  bodajże, czyli -7. Więc E z pierwszego =  $5-7=-2$ .  $E2=10-7=3$ . Czyli masz dodawanie

$-2^{(-2)} * 1,101 + -2^{(3)} * 1,110$

Skalujemy do większej czyli możemy przesunąć przecinek:

$-2^{(-2)} * 1,101 + -2^{3} * 1,110 = -2^{(-5)} * 1,101 * (-2)^3 + (-2)^3 * 1,110 = 0,00001101 * (-2)^{(3)} + (-2)^3 * 1,110 ?$

I tu już widać, że skoro mamy zapisać na 3 bitach, to wynik dodawania tego nie zmieni czyli wynikiem będzie liczba większa  $(-2)^3 * 1,110$  ?

OK - czyli ogólnie rzecz biorąc - "skalujemy" do większej liczby rozbijając to sobie tak jak wyżej?

Rozbiecie (-2) do większej (3).  $-5 + 3 = -2$ . LUZIK, ja to samo:D //rozumiem:)

A skąd to -5? Nagle? Aaa ok sorry ju z spie

// Przeskalować możesz do jakiej chcesz, ale najlepiej zrobić tak żeby mieć postać od razu 1,xxxx + 0,xxxxx , wtedy nie bedziesz musiał skalować drugi raz.

// Piszesz wtedy że wynikiem jest liczba większa i dopisujesz że jest utrata precyzji (loss precision).

// Jeżeli ktoś ma problemy z "przesuwaniem" przecinka przy skalowaniu to warto zapamiętać że jak mamy  $M * 2^E$  i zwiększamy E to wartość M musi się zmniejszyć i na odwrót.

//PRZYKŁAD:  $2^5 * 1,011$  chcemy przeskalać na  $2^{10}$ , więc przesuniemy mantysę o 5 w lewo:D czyli  $2^{10} * 0,00001011$

// Dokładnie ^^

Jakieś pomysły na 4 ?

Podobnie jak wyżej tyle że najpierw liczysz ile stron zajmuje proces, a potem ile takich stron się zmieści w 1024MB-(system)

Olewam te segmenty ? Bo tylko to mnie zmyliło. Ze stronami wiem jak sobie poradzić. Na co mi informacja o rozmiarze segmentu skoro na raz ładuję do pamięci i tak kilka stron dla kazdego procesu. Ale segment zajmuje  $(1+x)GB$  czyli cała pamięć LOL

Czy w takim wypadku nie będzie miejsca na procesy? Bedzie bo mamy segmentacje

stronicowana wiec wewnatrz segmentu 2GB jest iles tam stron po 8KB ale tylko połowa segmentu jest dostepna 1GB w tym 10MB zjada SO

Segmentowanie i stronicowanie moze byc stosowane jednocześnie.

Segmentowanie uzywasz do definiowania logicznych partycji podlegajacych kontroli dostepu a stronicowanie do zarzadzania przydzielaniem pamieci wewnatrz partycji.

k

218180

Zakladajac ze w eax mam 3 ( na poczatku, pierwsza linia)

Po wymnozeniu eax = 0x30000 czyli 192kb

Zbior roboczy powinien miec 192kb (bo  $2^{16} * 3$ )

Strona wychodzi 10 kb bo  $2 + k3$  KB =  $2 + 8$  KB 218180 h3=8

Wiec 20 stron na proces (8kb tracimy ze wzgledu na fragmentacje wewnętrzna, na ostatniej stronie tylko 2kb beda wykorzystane)  $192/10$  w góre =20

Potem tak jak mowiles  $(1024MB - \text{system})/10KB$  (zaokraglamy w dól) i tyle mozemy miec stron  $(1024MB - 10MB)/10kb = 1014 * 2^{10} / 10 * 2^{10} = 1014 * 2^{10} / 10 = 1014 * 1024 / 10 = 1038336/10 = 103833,6$  stron

$103833,6/20 = 5191$  po 20 na proces

// jakoś tak to było?

Potem patrze ile 20 (bo tyle stron proces potrzebuje) mieści się w tym

Czy zle myśle ?

Nie wiem co z tym rozmiarem segmentu miałbym zrobić

//jak ktoś wie czy to powyżej rozumowanie jest OK to niech pisze, bo ja już sam nic nie wiem

Może jeżeli kończy się miejsce w pamięci RAM to wtedy plik stronnicowy(?) zapisywany jest na dysku twardym. Takie rozwiązania są wykorzystywane, jeżeli pozwalamy na to w systemie.

^ To na górze wygląda przynwoicie... // ja bym tak zrobił na kole.

// a ja w sumie tak to machałem z pamięci i nie wiem czy dobrze, w sensie z tego co pamiętam jak ktoś robił przed skasowaniem

Było tak jak wyżej tylko ja nie robiłem obliczeń, ktoś dopisał to co na niebiesko jest Jeżeli sposób jest dobry... zagadka pozostaje : co zrobić z rozmiarem segmentu i czy wgl to ruszać, czy dla zmyłki dał ;p???????????????????

## Zadanie 5. 2015 termin 0

5. (5p) Podać definicję lokalności:  $h_0 \bmod 2 = 0$  – czasowej,  $h_0 \bmod 2 = 1$  - przestrzennej.

5. Lohelność adresowa - orzecza tendencję do powtarzania odwołań zrealizowanych w niedalekiej przeszłości.

Lohelność postrzenną - orzecza tendencję do odwoływań się do obiektów znajdujących się w obszarze adresowanym, w którym znajdują się obiekty, którymi zostały już wiele w programie.

## Koło 2014, TERMIN 0

Organizacja i Architektura Komputerów. Egzamin, termin 0. 17.06.2014  
Czas: 40 min. Używanie kalkulatorów: zabronione. Do notatek służy druga strona kartki. Przejrzysty zapis obliczeń ułatwia mi rozstrzyganie przypadków niejednoznacznych. Życzę Wam powodzenia – Piotr Patronik  
Imię i nazwisko: ..... Numer indeksu: .....  $h_5h_4h_3h_2h_1h_0$  Punkty: ...../25

1. (5p) Jest dany procesor o 4-bitowym słowie rozkazowym i poniższym kodowaniem rozkazów. Zapisać program w postaci mnemoników i podać wartości w rejestrach po wykonaniu 4 rozkazów (zapisanych szesnastkowo):  $h_3, h_2, h_1, h_0$ , zakładając, że wartości początkowe rejestrów wynosiły 0.

$ii\ v\ 0\ \text{ld}\ \$v,\ @ri\ \quad ri = v$   
 $ii\ j\ 1\ \text{add}\ \@ri,\ @rj\ \quad rj = ri+rj$

2. (4p) Narysować schemat sumatora prefiksowego  $(d_0\%3)+3$ -bitowego w architekturze  $b_0=0$  – Kogge-Stone'a,  $b_0=1$  – Sklansky'ego i przedstawić jego działanie dla dwóch dowolnie wybranych (różnych i niezerowych) wektorów wejściowych.

3. (6p) Są dane dwie liczby zmiennoprzecinkowe w standardzie podobnym do IEEE 754 z tą różnicą, że mantysa ma 7 bitów która jest szesnastkowo zapisana jako  $A=h_3h_2h_1h_0$  i  $B=h_1h_2h_3h_4$ . Obliczyć i podać ich sumy i ilorazy, tj  $A+B$  i  $A/B$ , wyniki podać dziesiętnie i szesnastkowo w formacie źródłowym. W przypadku nie-liczby, wykonać tylko operacje na mantysach.

4. (5p) Jest dany fragment kodu pewnego procesu. Niech rozkaz rand ładuje rejestr wartością zmiennej losowej wg rozkładu jednostajnego z przedziału od 0 do wartości z rejestru. Ile takich procesów można uruchomić, jeżeli rozmiar dostępnej pamięci głównej wynosi 1024MB?

```
mov $h_3, %eax
mov $0x100000, %ebx
mul %ebx

begin:
    mov %eax, %eax
    rand %eax
    mov %eax, (%eax)
loop begin
```

5. (5p) Podać definicję lokalności:  $b_0=0$  – czasowej,  $b_0=1$  - przestrzennej

## Koło 2014, TERMIN 0

## ROZWIĄZANIA:

### Zadanie 1. 2014 termin 0

1. (5p) Jest dany procesor o 4-bitowym słowie rozkazowym i poniższym kodowaniem rozkazów. Zapisać program w postaci mnemoników i podać wartości w rejestrach po wykonaniu 4 rozkazów (zapisanych szesnastkowo):  $h_3, h_2, h_1, h_0$  zakładając, że wartości początkowe rejestrów wynosiły 0.

```
ii v 0 ld $v, %ri    ri = v  
ii j 1 add %ri, %rj  rj = ri+rj
```

- 1) ii v 0      ld \$v, %ri      ri = v
- 2) ii j 1      add %ri, %rj    rj = ri+rj

#### Próba rozwiązania:

Osobiście nie rozumiem w pełni polecenia, ale widziałem próby rozwiązania tego zadania w następujący sposób:

założymy nr indeksu 200763

$h_3=0h=0000b$

$h_2=7h=0111b$

$h_1=6h=0110b$

$h_0=3h=0011b$

( kolejne bity traktujemy jako kolejne pozycje w tych schematach iiv0, iij1)

//A jak dokładnie wyliczane są te ii?  $h_2=7h=0111b$  czyli 2 starsze bity będą tym ii - 01? A potem to kolejny bit (tutaj 1) to będzie ta wartość j? I co potem z tym ostatnim bitem co zostaje oraz liczbą 1 (iij '1')?

// ostatni bit, który zostaje determinuje która instrukcja ma zostać wykonana.

jak 0 na końcu: ld

jak 1: add

//Oki, dzięki, a co z tą liczbą która jest w kodzie (po v jest to 0 a po j jest to 1)?

No przecież ci napisał właśnie.

//Myślałem że chodzi o ten ostatni bit z tej liczby  $7h=0111b$

No bo właśnie chodzi. Na podstawie tego ostatniego bitu w liczbie decydujesz czy to ma być ld czy add.

//Oki, teraz rozumiem :D

Jesteś pewien, że rozumiesz?

//0101b: ii = 01 v/j = 0? instrukcja add=1?

Nom. Tylko, możliwe, że jutro będzie trochę inaczej ale już jak wiadomo o co tu chodzi to będzie łatwiej zrozumieć. No chyba, że da identyczne.

**Rozkaz  $h_3$ : zero na końcu czyli instrukcja 1):**

$ii = 00, v=0$ , stąd rozkaz: ld \$0, %r0

$r0 = 0$

**rozkaz h2: jeden na końcu czyli instrukcja 2):**

ii = 01, j = 1 zatem rozkaz add %r1, %r1

r1 = 0

// - skąd wiadomo którą wartość 'i' wybrać? mamy tu 0 i 1 dla ii=01. Chyba, że jest to po prostu liczba zapisana w binarnym i dlatego mamy r1, a nie r0?

// - 00 -> 0, 01 -> 1, 10 -> 2, 11 -> 3

**rozkaz h1: zero - instrukcja 1):**

ii = 01, v = 1 zatem ld \$1, %r1

r1 = 1

**Rozkaz h0: jeden - instrukcja 2):**

ii=00, j=1, stąd rozkaz: add %r0, %r1

r0 = 0; r1 = 1

## Zadanie 2. 2014 termin 0

2. (4p) Narysować schemat sumatora prefiksowego  $(d_0 \% 3) + 3$ -bitowego w architekturze  $b_0=0$  – Kogge-Stone'a,  $b_0=1$  – Sklansky'ego i przedstawić jego działanie dla dwóch dowolnie wybranych (różnych i niezerowych) wektorów wejściowych.

Do obliczenia liczby poziomów:  $P = \log_2 n$ , gdzie  $n$  to liczba bitów, wynik podciągamy w górę, tzn.

np dla  $n=13$ ,  $p=\log_2 13=4$

Sklansky - oczywiście uciac do zadanej liczby bitow

//a dokładnie co oznacza ten zapis  $d_0 \% 3 + 3$  ? i co oznacza  $b_0 = 0$  ?

//  $d_0$  - najmniej znaczaca cyfra indeksu studenta

//  $\% 3$  - modulo 3 (reszta z dzielenia przez 3)

//  $+3$  - plus (dodac) 3

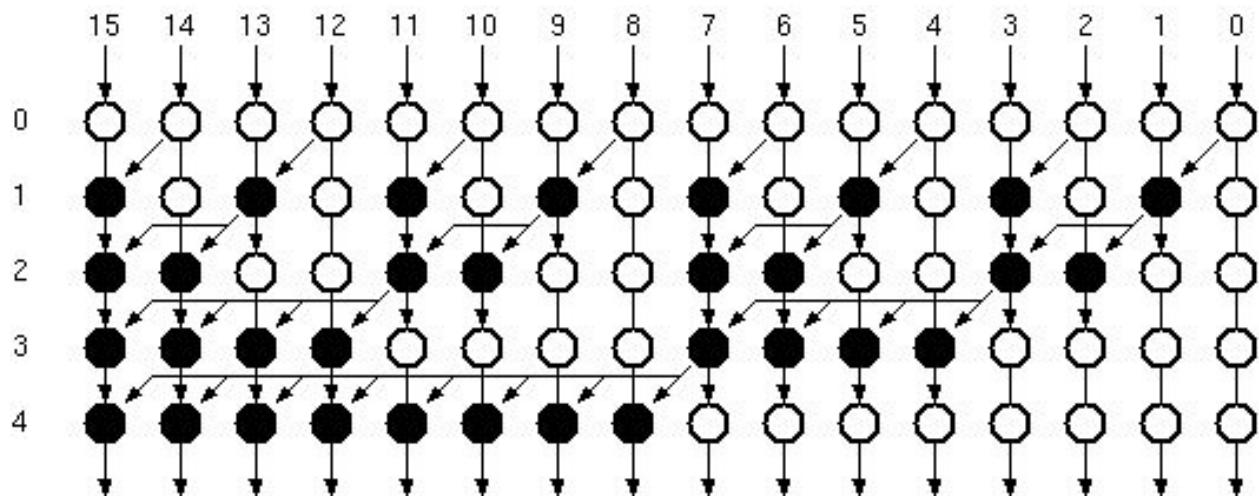
//  $b_0$  - najmniej znaczacy bit numeru indeksu studenta -  $b_0=0$  indeks parzysty,  $b_0=1$  nieparzysty //czyli gdy indeks kończy się na np. 0 to mamy narysować sumator 3-bitowy w arch. de\_Cobble'a Stone'a (WTF xD a nie koga-stona ? ZA DUZO MINECRAFTA)(jak juz to kogga-stone'a)(jak już to kogge'a-stone'a)(de\_cbble & stone'a). czyli z tego rysunku poniżej mają zostać tylko kolumny 0, 1 i 2 ?

// chyba tak

//czyli jak mam log o podstawie 2 z 3 (=2) to mam poziomy 0,1 czy 0,1,2? #HELP \_\_\_\_\_

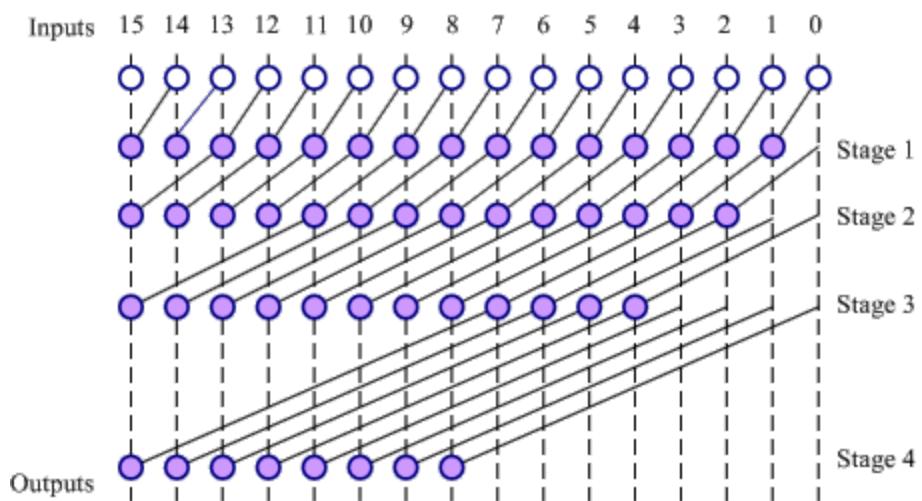
//pojżej mamy 16sto bitowy, czyli  $\log_2(16)$  to 4, co daje nam poziomy 0,1,2,3,4... (czyli w twoim przypadku bd 0,1,2).

**Sklansky:**



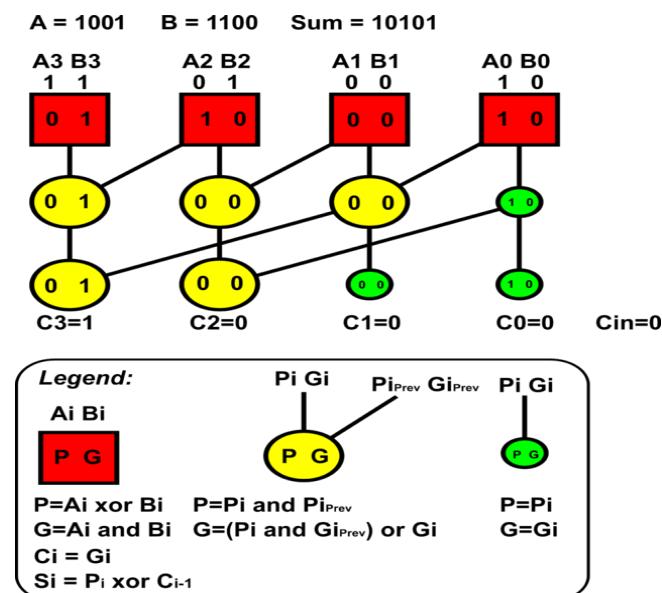
//na pewno tak jest, że się po prostu obcina kolumny - tak

**de\_Kobble-Stone:**



a jak pokazac przykładowe działanie tego na dowolnych wektorach?

A no np tak:(wzory się zgadzaja, przerobiłem parę przykładów, dla różnych dł. słów wejściowych)



Dla Sklanskiego wzory zostają takie +same, tylko inny schemat.

Ostatnia suma jaka należy obliczyć(dla powyższego przykładu) to S3, a S4 to po prostu przeniesienie z poprzedniej pozycji.

czy w ostatnim elemencie musimy dodać kreskę w przód ? tzn ostatnie przeniesienie ? w przykładzie wyżej na najstarszej pozycji sumujesz 1 + 1 i tracisz wynik bo na tej pozycji zostaje 0 a 1 ma przejść na następną. Jak nie ma kolejnego wyjścia na starszą pozycję to tracisz najsatrzszą pozycję.

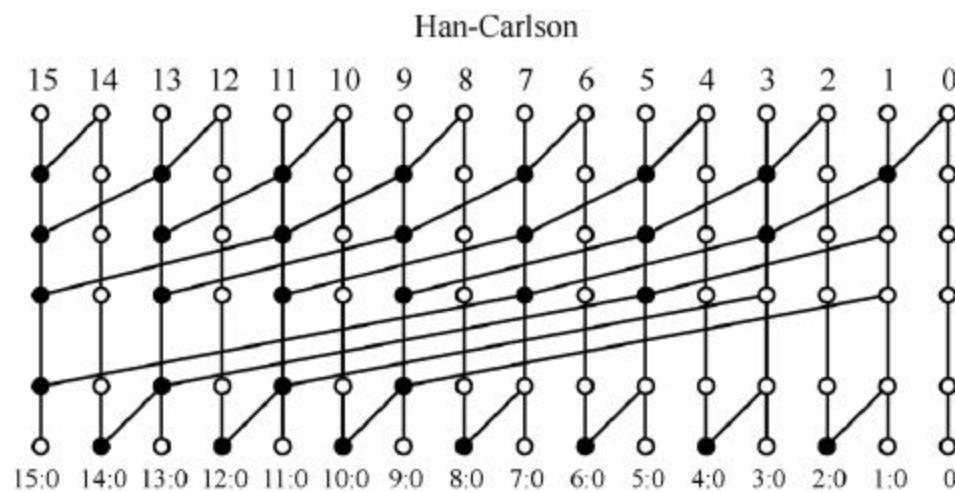
Ogólnie tak: oprócz powyższego znalazłem jeszcze inny schemat z obliczonym przykładem i również nie miał on żadnej kreski przy ostatnim elemencie. Teoretycznie można dodac taką kreskę, w praktyce nie uważam tego za konieczne, dość intuicyjne jest że na ostatnią pozycję wyskoczy nam przeniesienie z poprzedniej.

Zamiast kreski najlepiej napisać np. "overflow=1" i tyle.

//gdyby ktoś się zastanawiał, to przy wzorze na sumę Pi jest P pochodzący z czerwonego (poprzedniego - Gprev i Pprev to odpowiednio generacja i propagacja z POPRZEDNIEJ pozycji) kwadratu a nie z góry (na rysunku z dołu bo rośnie w dół) drzewa jak G w przypadku Ci

WZORKI do CARLSONA takie same - nie prawda,  $l_{\text{poziomow}} = \log_2 n + 1$ , zawsze dochodzi ten nieregularny rząd który widać na samym dole poniższego obrazka

U GORY JEST ROZWIAZANIE DLA 3 4 5 bitow



### Zadanie 3. 2014 termin 0

3. (6p) Są dane dwie liczby zmiennoprzecinkowe w standardzie podobnym do IEEE 754 z tą różnicą, że mantysa ma 7 bitów która jest szesnastkowo zapisana jako  $A=h_3 h_2 h_1 h_0$  i  $B=h_1 h_2 h_3 h_4$ . Obliczyć i podać ich sumy i ilorazy, tj  $A+B$  i  $A/B$ , wyniki podać dziesiętnie i szesnastkowo w formacie źródłowym. W przypadku nie-liczby, wykonać tylko operacje na mantysach.

//czyli 1 bit to znak, 8 bitów to wykładnik i 7 bitów to mantysa.

Dla np. 198691

	Z	E	M	
A:	8691	1	000 0110 0	001 1001
B:	9689	1	001 0110 1	000 1001

to w ogóle jest dobrze?// dwie ostatnie nie powinny być odwrotnie?

Skorzystajmy z metody dzielenia nieodtwarzającego

$$\begin{array}{r} (0)1101111 | \quad (0)1110111 \\ \underline{(0)} | \\ (1)0010001 | \quad : (1)0001001 \\ + (0)1110111 \\ \hline (0)0001000 \end{array}$$

poddaje się to się robi dotad az sa liczby za skala a tu nie ma zdanych xd

Dlaczego tutaj uzupełnienie 1? Przecież to że liczba ujemna, nie musimy w to mieszać dzielenia mantys.

Z- znak

E- wykładnik

M- Mantysa / Po tomczakowemu mnożnik (w g. Tomczaka określenia mantysa sie juz nie używa )

Tutaj chyba będzie po prostu B, A jest mniejsze od B  $2^{31}$ -krotnie, więc A nie ma żadnego wpływu. Mam rację?

Schematy ogólne:

**Dzielenie** - dzielimy mantisy odejmujemy wykładniki i najprawdopodobniej dodajemy do wykładnika tyle ile trzeba było przesunac przecinek aby wyrownac mantysę wynikową zeby byla postaci 1,1010101

**Mnożenie** mnozymy mantisy dodajemy wykładniki i jak wyżej dalszy ciąg

W obydwu przypadkach pamiętamy o zamianie przy operacji dodawania i odejmowania na  $+N$  czyli negujemy wszystkie bity oprócz najstarszego wykonujemy operacje wynik zamieniamy z powrotem na podstawie 2 czyli negujemy wszystkie bity oprócz najstarszego poraz kolejny.

Przy dodawaniu wyrównania juz nie zamieniamy na  $+N$  tylko dodajemy w dwójkowym

Odwrotność - zamieniamy tak jak wyżej z  $+N$  na 2 czy na odwrot znaczy tak jak wyżej opisane jest. Po zamianie obliczamy odwrotnosc liczby to chyba kazdy wie jak w u2 zrobic  
Znowu zamieniamy na  $+N$

Dzielimy 1 przez mantysę przesuwamy przecinek aby liczba bitów licznika i mianownika byla taka sama. Przesuwamy przecinek w prawo do pierwszej jedynki. Na koniec odejmujemy od wykładnika tyle ile trzeba było przesunąć.

### A potrafi ktoś zrobić dodawanie?

W dodawaniu to się normalizuje do liczby wiekszej, bo ta mniejsza jest mniej istotna.

Wzór jest taki że  $M1 \cdot 2^E1 + M2 \cdot 2^E2 = 2^E2(M1 \cdot 2^{E-E2} + M1)$  przy założeniu że E1 jest wieksze niż E2. Zaczepniete z ćwiczeń u Postawki.

Wzór to wiem, też mam nawet od niej notatki. Mam przykład z ćwiczeń, gdzie E2 jest mniejsze od E1 dokładnie o 3 i wtedy po prostu mnożymy mantysę drugiej liczby (wraz z ukrytą jedynką) razy  $2^{-3}$ , ale w przypadku mojego nru indeksu z dzisiejszego koła dostaje liczbę o 32 mniejszą, to co,  $2^{-32}$ ? Chyba nie za bardzo o tyle przesuwać w lewo.

Nawiązując do powyższego problemu - przesunięcie o 32 w lewo/prawo, dodanie i powrót do postaci bez przesunięcia nie ma prawa zmienić wyniku który jest zapisany na 8 bitach(zmiana będzie obserwowana dopiero przy 32 bicie, a po powrocie i zachowaniu precyzji będzie całkowicie niewidoczna). Wobec tego zamiast robić przesunięcie i pisać 30 bitowe liczby można słownie opisać cały proces i wynik takiego dodawania napisać w kolejnej linijce.

OK, to przykładowo może ktoś rozpisze dodanie takich dwóch liczb?

0 00001110 0010101

0 00101110 0000000

S wykładnik mantysa

Jak te wykładniki się wyrównuje, mają być całkowicie sobie równe (tzn odpowiednie bity mają się zgadzać?). - chyba tak, w sumie napewno tak || no dobra, akurat w podanym przykładzie wykładniki różnią się tylko jednym bitem, na pozycji 5., czyli różnią się o 32. Jak to wyskalować?

Masz pierwszą liczbę:  $1,0010101 \cdot 2^{-113}$  i drugą  $1,0000000 \cdot 2^{-81}$

Robisz tak:  $1,0010101 \cdot 2^{-32} \cdot 2^{-81}$  z pierwszą i masz

$0,0\ldots010010101 \cdot 2^{-81}$  (jakoś tak) - Nie zmieni nam to więc wyniku, bo mantysę mamy zapisać tylko na 7 bitach

dalej je sumujesz i wychodzi ci  $1,0000000 \cdot 2^{-81}$  i koniec - **tak mi się wydaje, poprawcie mnie jak źle**

**Wykładnik jest zapisany w postaci  $+N$ , chyba, sam nie wiem już**

Zgadza się, ale jakbyśmy chcieli zapisać wynik dwójkowo, to do wykładnika -81 dodajemy obciążenie? -  $81 + 127 = 46$ , i to jest wykładnik wynikowy? Czy liczba wynikowa będzie ujemna (DODATNIA - przecież sumował dwie dodatnie liczby) (ze znakiem 4-0) i wykładnikiem o wartości 46?

Nom na to wygląda. Podczas sumowania dwóch liczb gdy jedna jest dużo mniejsza od drugiej to ta mniejsza nie ma żadnego wpływu. Pozostaje po prostu ta większa liczba. Czyli wynik będzie taki sam jak ta druga liczba.

## Zadanie 4. 2014 termin 0

4. (5p) Jest dany fragment kodu pewnego procesu. Niech rozkaz rand ładuje rejestr wartością zmiennej losowej wg rozkładu jednostajnego z przedziału od 0 do wartości z rejestru. Ile takich procesów można uruchomić, jeżeli rozmiar dostępnej pamięci głównej wynosi 1024MB?

```
mov $h_3, %eax  
mov $0x100000, %ebx  
mul %ebx  
  
begin:  
    mov %ebx, %eax  
    rand %eax  
    mov %eax, (%eax)  
loop begin
```

Jest dany fragment kodu procesu. Rand to funkcja generująca wartości od 0 do wartości w rejestrze. Ile procesów można uruchomić, jeśli rozmiar pamięci głównej to 1024 MB.

### **POPRAWIONE NK SPRAWDZI!!!!**

Dla indeksu: #####2 ->  $h_0 = 2$

```
mov $h_0, %eax      // eax = 23  
mov $0x100000, %ebx  
mul %ebx
```

```
begin:  
    mov %ebx, %eax  
    rand %eax  
    mov %eax,(%eax)  
loop begin
```

#### Rozwiązanie:

eax = 23

$0x100000 = 2^{10} \text{ kB} = 1\text{MB}$  // a nie  $2^{16}$  ???  $0x100000$  to szesnastkowo przeciez  $16^4 = 2^{16}$

Po wykonaniu mnożenia: ebx = 1MB \* 23 = 23MB //  ~~$2^{16} * 23 = 2^6 * 23\text{kB} = 1472\text{ kB}$~~

W pętli eax=ebx => eax=23MB

Ilość procesów:  $1024/23 = 44$  (floor)

## Zadanie 5. 2014 termin 0

Podaj definicję lokalności  $b_0=0$  - czasowej,  $b_1=1$  - przestrzennej

5. (5p) Podać definicję lokalności:  $b_0=0$  – czasowej,  $b_1=1$  – przestrzennej

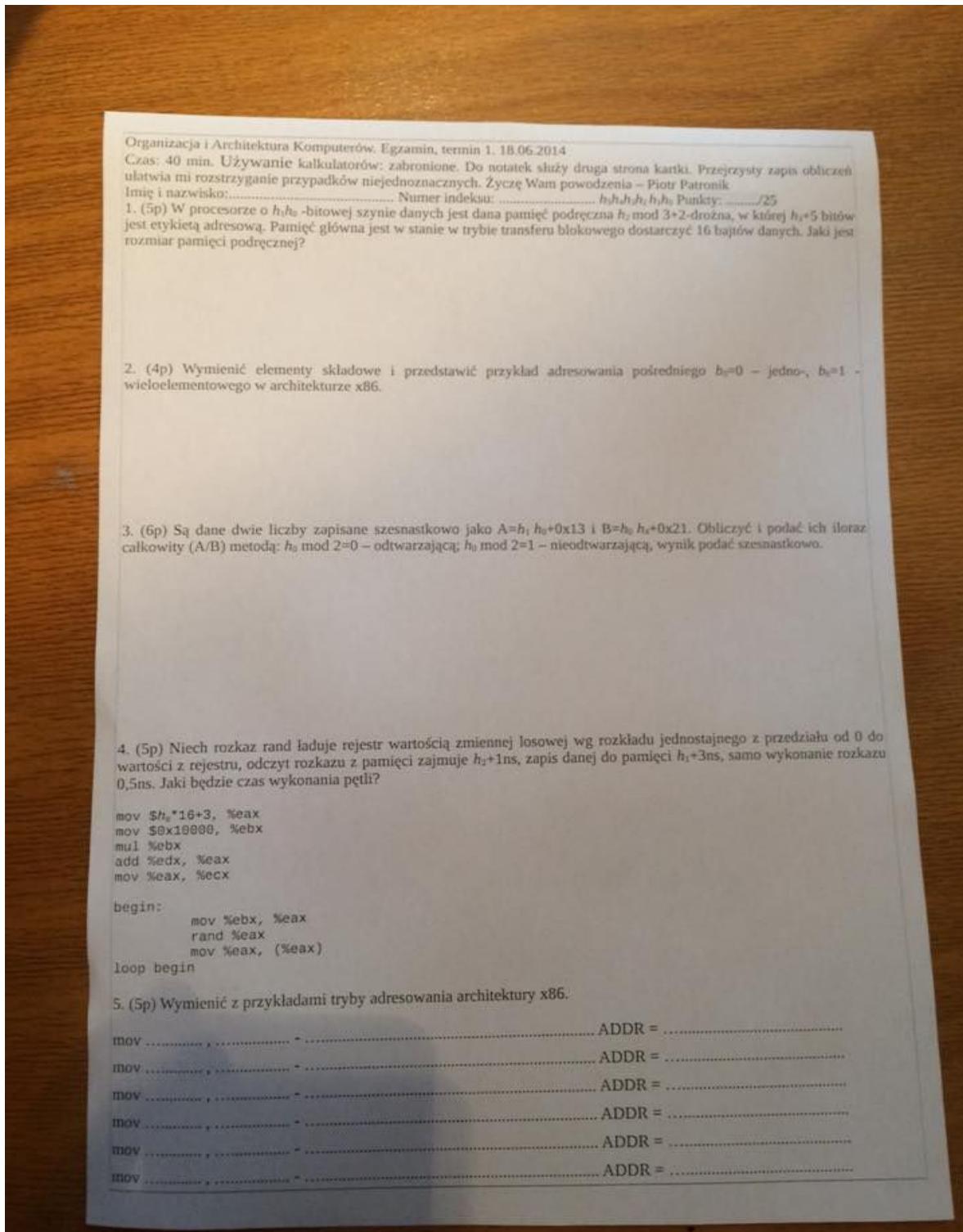
1. Lokalność czasowa oznacza tendencje do powtarzania odwołań realizowanych w niedawnej przeszłości.

2. Lokalność przestrzenna oznacza tendencje do odwołań do obiektów umieszczonych w obszarze adresowym obejmującym obiekty, które były już użyte w programie.

5. Lokalność czasowa – oznacza tendencję do powtarzania odwołań realizowanych w niedawnej przeszłości.

Lokalność przestrzenna – oznacza tendencję do odwołań do obiektów znajdujących się w obszarze adresowym, w którym znajdują się obiekty, który zostaly już włączone w programie.

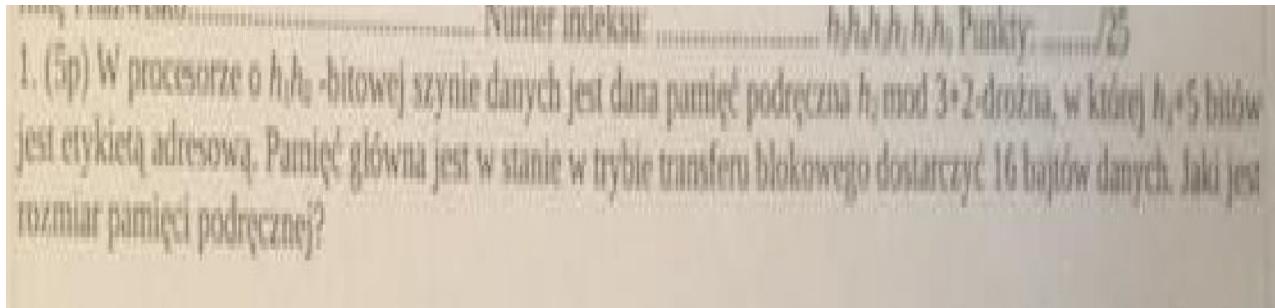
# Koło 2014, TERMIN 1



# Koło 2014, TERMIN 1

## ROZWIĄZANIA:

### Zadanie 1. 2014 termin 1



Co patron kazał dopisać w tym zadaniu? Z tego co pamiętam kazał zmienić coś w szynie bitowej.  
Nie miało to być +32 ?

// Miałobyć 32 + h1

To zadanie jest bardzo podobne do zadań które dawał biernat na swoim kole.

<https://docs.google.com/document/d/1ruzpUn0eqahelJ-hvL0By1m7mYF16ZyiEb06DIls5nxU/edit#>  
strona ~ 16 . Jednak nie bardzo wiem jak to poskładać - my mamy mniej danych

Rozwiążane na podstawie:

- <http://www.d.umn.edu/~gshute/arch/cache-addressing.xhtml> -> An example
- <http://www.eecg.toronto.edu/~moshovos/ECE243-07/I26-caches.html> -> Example #2

TAG adres linii	SET INDEX indeks linii SETU?	BLOCK OFFSET indeks słowa
--------------------	---------------------------------	------------------------------

Postać adresu w RAM wg <http://www.d.umn.edu/~gshute/arch/cache-addressing.xhtml> (nazwy po polsku wg slajdów JB)

Szerokość adresu = rozmiar szyny.

Niech szyna ma 38 bitów, cache jest 4-drożne a liczba bitów etykiety wynosi 14. Wiemy, że w trybie blokowym pamięć przesyła  $2^4 = 16$  (**adresacja jest bajtowa**) bajtów, czyli rozmiar bloku(BLOCK OFFSET) to 4.

Adres linii (TAG)(czyli rozmiar etykiety adresu) to 14 bitów .

Zostaje 20 na SET INDEX.

Nawiązując do linku:

$$\text{liczba setów cache} = \frac{\text{Rozmiar cache}}{\text{Ilosc bajtów w secie}}$$

+`Znając ilość bajtów w secie (4 drogi \* 16 bajtów = 64) oraz ilość setów (maksymalna wartość set index)  $2^{20}$  możemy wyliczyć rozmiar cache jako iloczyn ilości setów cache i ilości bajtów w wierszu. Mamy zatem:

$$2^{20} * 64 \sim 64 MB$$

# Ja widziałbym to zdanie tak:

# Przykładowy indeks 200000 // tutaj liczby nie mają aż takiego znaczenia

**linia\_cache = 16B - w poleceniu podane 16 bajtów**

**ilosc\_bitow\_na\_szynie = 32b** // szyna skąd to 32? // w zależności od procesora podanego w zadaniu (chyba) ? //bo właśnie jest w procesorze o h1h0 bitowej szynie i nie widzę gdzie to wykorzystano

# przeczytaj co jest napisane wyżej, patron powiedział żeby zmienić na h1h0 + 32, dla samego h1h0 nie miałoby sensu jak ktoś ma indeks kończący się 00

**Etykieta = 5**

Etykieta = h0+5 = 5 w tym przypadku,

Offset =  $\log_2(16) = 4$

offset =  $\log_2(\text{linia\_cache})=4$  **okey**

set = szyna - offset - etykieta = 23bity

liczba\_setów =  $2^{\text{set}} = 2^{23}$  (nie jest to po prostu liczba setów?) blok to to samo co linia

“ Each set contains 2 cache blocks (2-way associative) so a set contains 32 bytes.”

zrodlo:<http://www.d.umn.edu/~gshute/arch/cache-addressing.xhtml> - > to tlk przykład

Generalnie cache podzielony jest na bloki/linie np po 16 bajtów kazda linie adresujemy ofsetem. Dodatkowo cache podzielony jest na sety (set associative) (PTM)

<http://wiki.fl9.eu/pwr/ptm/cache-budowa-i-sposoby-organizacji>

w każdym secie jest kilka lini(droznosc/ ways). Gdy jest jeden set (fully associative) to każdy rekord może być w każdym miejscu seta. Gdy każdy set ma 1 element(way/linia) to jest direct mapped cash\$\$

liczba\_lini = liczba\_setów \* drozszosc =  $2^{23} * 2 = 2^{24}$

rozmiar\_lini = 16B =  $2^4 B$

wielkosc\_cache = liczba\_lini \* rozmiar\_lini =  $2^{24} * 2^4 = 2^{28} = 2^8 MB = 256 MB$

# Nie dam sobie uciąć ręki za powyższe obliczenia, dobrze byłoby jakby ktoś to potwierdził  
// 5/5 I termin 2016 według sposobu powyżej

Zrobiłem wg 2 sposobów - wyszło to samo, więc chyba jest OK :)

Panie i Panowie: <http://157.158.56.13/diplomy/1/sposoby4.html> polecam  
przeczytac(2016)

Wynik ma sens, ale czy tam robi jakąś różnicę, że linia jest w BAJTACH a cała reszta danych jest w bitach? //w bitach podane są części adresu który jest potrzebny do generowania trafień/chybień, a w bajtach rozmiar linii/wielkosc cache'a

//@UP. Nie robi. Ważne żeby wszędzie (w każdym równaniu) jednostki się zgadzały, a w powyższym przykładzie tak właśnie jest. Dzięki!

$$1. \quad h_1 h_0 + 32 = 45 + 32 = 77 \text{ bitowa szyna danych (średniość adresu)}$$

teg (etykieta) set index (index linii) block offset (index stawu)

$$\text{drożność} = h_2 \% 3 + 2 = 2$$

$$\text{etykieta} = h_3 + 5b = 11b$$

$$\text{rozmiar\_linii} = 16B$$

$$\text{offset} = \log_2 16 = 4$$

$$\text{set} = 77 - 11 - 4 = 62$$

$$\text{liczba\_bloków} = 2^{\text{set}} = 2^{62}$$

$$\text{liczba\_linii} = \text{liczba\_bloków} \cdot \text{drożność} = 2^{62} \cdot 2 = 2^{63}$$

$$\text{rozmiar\_cache} = \text{rozmiar\_linii} \cdot \text{liczba\_linii} = 16 \cdot 2^{63} [B] =$$

$$= 2^{67} [B] = 2^{37} [GB]$$

## Zadanie 2. 2014 termin 1

2. (4p) Wymienić elementy składowe i przedstawić przykład adresowania pośredniego b=0 - jedno-, b>1 - wieloelementowego w architekturze x86.

rozwinać/zweryfikować, ja to wypisałem z książki Biernata

Pośrednie jednoelementowe - //07-ak2-wyklad7 od 15 slajdu

Elementy:

- Operand
- adres docelowy //Zgadzam się Przykłady:
  -
- bezwzględne pośrednie
- rejestrów pośrednie
- rejestrów pośrednie z modyfikacją

Pośrednie wieloelementowe - //07-ak2-wyklad7 od 17 slajdu

Elementy:

- baza (adres odniesienia w rejestrze procesora; wskaźnik adresu bloku)
- przemieszczenie bazy (stała adresowa, w adresowaniu dwupoziomowym dodawana do bazy, stanowiąca osobne słowo rozszerzenia kodu rozkazu; adres odniesienia (wskaźnika) bloku)
- indeks (umieszczony w rejestrze procesora wraz z mnożnikiem skali wskazanym w słowie lub słowach rozszerzenia kodu; bieżący wskaźnik w bloku)
- Skala indeksu - (mnożnik indeksu wskazany w słowie kodu)
- relokacja (stała dodawana do wyznaczonego adresu, stanowiąca osobne słowo rozszerzenia kodu rozkazu)

To jak to jest

W trybie mov a(%ebx,%edi,0x02) mamy:

a jako offset/przemieszczenie bazy  
ebx baza  
Edi indeks  
0x02 skala

Gdzie jest relokacja?

Jest inny tryb adresowania?

Przykłady (dwuelementowe):

- bazowe z przemieszczeniem
- indeksowe z przemieszczeniem
- bazowo-indeksowe
- względne z przemieszczeniem
- względne indeksowe

adresowanie wieloelementowe jednopoziomowe

adresowanie wieloelementowe dwupoziomowe

2.

adresowanie:

- bezpośrednie:

- zeroelementowe

- np. zwarte: mul al

- operend domniemany - eax

- jednoelementowe

- np. rejestrowe bezpośredni mov %eax, %ebx

- argument w rejestrze

- pośrednie:

- jednoelementowe

- np. rejestrowe pośrednie mov (%eax), %ebx

- adres efektu deng' w rejestrze

- wieloelementowe

- obliczenie wskaźnika na podstawie słówek

---

mov al(%ebx, %edi, 2), %eax

(14)

al - pneumatyzowanie bazy (stos, adres odniesienia)

ebx - zawiera wskaźnik adresu bloku (baza)

edi - zawiera bieżący wskaźnik w bloku (indeks)

2 - skala, mnożnik indeksu

## Zadanie 3. 2014 termin 1

3. (6p) Są dane dwie liczby zapisane szesnastkowo jako A=h<sub>1</sub> h<sub>0</sub>+0x13 i B=h<sub>1</sub> h<sub>0</sub>+0x21. Obliczyć i podać ich iloraz całkowity (A/B) metodą: h<sub>1</sub> mod 2=0 – odtwarzającą, h<sub>1</sub> mod 2=1 – nieodtwarzającą, wynik podać szesnastkowo.

Dzielenie nieodtwarzające w systemie uzupełnieniowym (ogólne przypomnienie)

[http://www.pwrhelp.za.pl/ak1/dzielenieUzup\\_v11.pdf](http://www.pwrhelp.za.pl/ak1/dzielenieUzup_v11.pdf)

Ma ktoś jakieś materiały do przypomnienia odtwarzającego?

<http://sendfile.es/pokaz/399860---pB4b.html>

Ile tam jest na fotce w B? H0h1 czy h0h4 czy h0hA ?? Pyt. 2016

**PYTANIE:**

**O co chodzi z zapisem tej liczby w formacie h1h0+0x21. Jeżeli np h1=4 a h0=7, to h1h0 to liczba 47 (dziesiętnie), którą trzeba przekonwertować na system 16-tykowy, a następnie dodać do 0x021, żeby uzyskać liczbę końcową?**

//A to nie jest tak ze 47 to juz jest w hex ? i dodajemy do tego 0x21. To bedzie 68(hex) tak ?  
//Tak, 47 jest już w hex, dodajemy 21 i mamy 68. Zawsze można też zapytać Patronika na kole. tez tak myśle

Może chodzi o to, że gdy h1=4 i h0=7:

A=h1 h0+0x13 to tylko do h0 dodajemy 0x13?

//Nie , ja na kole zrobilem tylko to zadanie + jakieś adresowanie i mialem 10 pkt wiec bedzie tak jak wyzej pisza // w jaki sposób je rozwiązałeś? przekształciłeś na NB/U2 ?

p/z hex na binarke, rozszerzenie (0) i dziele obie

**PROSZĘ O SPRAWDZENIE** poniższego rachunku, coś mi się nie zgadza, zakładając, że zamieniamy liczbę z hex na u2 to mamy liczbę ujemną przez dodatnią i wynik powinien wyjść ujemny, w takim razie co jest nie tak z pierwszym działaniem ?

87 : 68 // czy 87 to liczba ujemna ????? - NIE!

Dzielenie jest źle, skoro 87 jest liczbą dodatnią to powinna wyglądać tak:

(0)10000111 O ile te liczby tak właśnie wyglądają // racja

$$\begin{array}{r} (1)0000111 : (0)1101000 \quad \text{znaki niezgodne +} \\ +01101000 \\ \hline 111011110 \quad \text{niezgodne +} \quad q_0=0 \\ + \underline{01101000} \\ \hline 010001100 \quad \text{zgodne -} \quad q_1=1 \\ - \underline{01101000} \\ \hline 01001000 \quad \text{zgodne -} \quad q_2=1 \\ - \underline{01101000} \\ \hline 11000000 \quad \text{niezgodne +} \quad q_3=0 \\ + \underline{01101000} \end{array}$$

01010000	zgodne -	q4=1
<u>01101000</u>		
11010000	niezgodne -	q5=0
<u>- 01101000</u>		
5 01110000	zgodne -	q6=1
= <u>01101000</u>		
0001000	zgodne -	q7=1

Wynik 01101011 = 6B // to dzielenie jest poprawne, ale to nie 87 i 68

Istnieje kalkulator obliczający dzielenie w u2:

: 
Start

X* D > 0	1011111001 100111000 010100110
+(- D)	<u>010000111</u> : 01101000 10011000
+(+ D)	1110110111 <u>01101000</u>
+(- D)	000111110 <u>10011000</u>
+(+ D)	1110101100 <u>01101000</u>
+(- D)	000101000 <u>10011000</u>
+(+ D)	1110000000 <u>01101000</u>
+(- D)	0001110000 <u>10011000</u>
+(+ D)	0000100000 <u>10011000</u>
+(- D)	1101010000 <u>01101000</u>
+(+ D)	11011110000

Dopisanie znaku na końcu reszty: 0

nawigacja

Do przodu

Kalkulator: [http://speedy.sh/tJ7s3/dzielenie-On\\_twarzajace.jar](http://speedy.sh/tJ7s3/dzielenie-On_twarzajace.jar) / dzięki!

Jeżeli komuś udało się pobrać ten plik z tego gównianego hostinu to czy może wrzucić to na jakiś normalny hosting?

$$3. A = h_1h_0 + 0 \times 13 = 45_{16} + 13_{16} = 58_{16} = 01011000_2$$

$$B = h_0h_1 + 0 \times 21 = 54_{16} + 21_{16} = 75_{16} = 01110101_2$$

m. odtwarzanie

$$\begin{array}{r}
 & & 10001011 \\
 & \underline{01100000} & : - 01110101_2 \\
 \underline{01011000}_2 & & \\
 + & \underline{01} & \\
 \underline{01011000} & & \\
 + & \underline{10001011} & -D \\
 \underline{0011110110} & & \\
 + & \underline{10001011} & -D \\
 \underline{000000010} & & \\
 + & \underline{10001011} & -D \\
 \underline{100011101} & & \\
 + & \underline{01110101} & +D \\
 \underline{000000100} & & \\
 + & \underline{10001011} & -D \\
 \underline{100011101} & & \\
 + & \underline{01110101} & +D \\
 \underline{000000100} & & \\
 + & \underline{10001011} & -D \\
 \underline{100011101} & & \\
 + & \underline{01110101} & +D \\
 \underline{000000000} & & \\
 + & \underline{10001011} & -D \\
 \underline{11001011} & &
 \end{array}$$

$A/B \approx 0, C_0, 0_{16}$

m. nieodtwierj. cd

$$\begin{array}{r} 10001011 \\ \underline{011000000} \\ 01011000_2 \quad : \quad 01110101_2 \\ + 10001011 \quad -D \\ \hline 111000110 \\ + \underline{01110101} \quad -D \\ \hline 01110110 \\ + 10001011 \quad -D \\ \hline 000000010 \\ + \underline{10001011} \quad -D \\ \hline 100011010 \\ + \underline{01110101} \quad -D \\ \hline 101001110 \\ + \underline{01110101} \quad -D \\ \hline 100110110 \\ + \underline{01110101} \quad -D \\ \hline 101101010 \\ + \underline{01110101} \quad -D \\ \hline 11001011 \end{array}$$

$$A/B \in \mathbb{Q}, C, D \in \mathbb{Z}$$

## Zadanie 4. 2014 termin 1

4. (5p) Niech rozkaz rand ładuje rejestr wartością zmiennej losowej wg rozkładu jednostajnego z przedziału od 0 do wartości z rejestru, odczyt rozkazu z pamięci zajmuje  $h_1 + 1\text{ns}$ , zapis danej do pamięci  $h_1 + 3\text{ns}$ , samo wykonanie rozkazu  $0,5\text{ns}$ . Jaki będzie czas wykonania pętli?

```
mov $h,*16+3, %eax
mov $0x10000, %ebx
mul %ebx
add %edx, %eax
mov %eax, %ecx

begin:
    mov %ebx, %eax
    rand %eax
    mov %eax, (%eax)
loop begin
```

W zadaniu ważne jest poprawne określenie gdzie zaczyna się pętla i gdzie kończy. W skład jednej iteracji pętli wchodzą w tym przypadku cztery instrukcje:

1. mov %ebx, %eax
2. rand %eax
3. mov %eax, (%eax)
4. loop Begin

Po wyliczeniu czasów konkretnych operacji można łatwo obliczyć czas wykonania jednej iteracji pętli; ja zakładam, że: odczyt **rozkazu** AP = 9ns, zapis do rejestru (Zapis danej do pamięci????)= 8ns, wykonanie rozkazu 0,5ns.

// moze ktos to wytłumaczyc dla jakiegos nr indeksu? I po kolei jakie operacje składaja sie na dana instrukcje ;)

1.  $9+0,5 = 9,5\text{ns}$  czemu nie ma tu odczyt ROZKAZU +zapis +wykonanie //bo nie ma zapisu m

do pamieci // a gdzie tu niby jest odczyt z PAMIĘCI? 1 instrukcja przenosi tylko zawartosc rejestru ebx do eax, to nie jest adresowanie pośrednie....

2.  $9+0,5 = 9,5\text{ns}$  i tu jak wyzej

3.  $9+8+0,5 = 17,5\text{ns}$  9 odczyt rozkazu 0,5 wykonanie 8 zapis

4.  $9+0,5 = 9,5\text{ns}$  i tu? Przeciez tu jest cmp i decremntacja, wiec powinno byc wiecej?  
1 rozkaz ocztyujesz z pamieci i wykonujesz to ze programista robi go za pomoca kilku instrukcji nie znaczy ze architektura nie ma jakiegos mechanizmu zeby to zrobic szybko w tej komendzie :)

RAZEM: 46ns

Kolejnym etapem jest określenie ilości iteracji. W instrukcji 1. do %eax ładuje wartość w zależności od indeksu (u mnie \$83); w 2. Zamieniamy z hex dec żeby się lepiej liczyło -  $0x10000_{(\text{HEX})} = 2^{16} = 65536_{(\text{DEC})}$ . Następnie mnożymy wartości i otrzymujemy  $5\ 439\ 488_{(\text{DEC})}$  i ta liczba mieści się w rejestrze 32-bit, więc trafi w całości do rejestrów %eax (wynik mnożenia %edx -

starsza część wyniku, %eax – młodsza). Po wykonaniu 3. instrukcji w %eax będziemy mieli tę samą wartość (5 439 488). Instrukcja 4. skopiuje wartość w %eax do %ecx (to będzie ilość iteracji pętli).

Mamy już wszystko co potrzeba do obliczenia czasu wykonywania pętli. Należy również wiedzieć jak działa instrukcja loop. Otóż robi ona cmp \$0, %ecx i jeśli wartość jest większa od 0 to wykonuje skok do etykiety (tutaj begin).

Czas wykonywania petli wynosi zatem: **czas iteracji \* ilość iteracji**, czyli **46\*5439488 = 250 216 448 ns**.

//Dlaczego operację : **mov %ebx, %eax** nie traktujesz jako: odczyt+ **zapis do rejestru + wyk. rozkazu?**

//Mariusz(nie Banach!): Bo zapis to zapis do rejestru a nie do pamięci (tak mi się wydaje :P) adresowanie bezpośrednie, jakby było posrednie to (%eax) a jak bazowo indeksowe to już powinno być jeszcze wiecej np mov a(%ebx,%edi,2), %eax

// Patryk : loop dekrementuje ecx więc chyba też powinien być odczyt i zapis, ale nie jestem pewien. Rand według patronika “ładuję” więc chyba sam zapis.

//Mariusz: Pytałem patronika na kole i mówił, że rozpatrujemy to na poziomie kodu.

// Patryk: czyli według niego jeśli nie mamy konkretnego zapisu w kodzie to nie bierzemy pod uwagę. Heh bez sensu, bo instrukcje sobie a on sobie :D

//Mariusz: TAK! To jest gość z ZAKa. Jego mózg pracuje na “Higher level of abstraction” :D

//Piotrek: czyli fakt, że przy instrukcji **mov %eax, (%eax)** prawdopodobnie program się wykrzaczy Bo będziemy próbowali zapisać gdzieś w pamięci, gdzie może nam nie wolno, nie ma znaczenia? Chyba mi mignęła już jakaś dyskusja na ten temat, ale nie mogę odnaleźć.

4.

$$\text{odległość} = h_2 + 1 \text{ [ns]} = 0 \cdot 1 = 1 \text{ [ns]}$$

$$\text{zapis-do-pamięci} = h_1 + 3 \text{ [ns]} = 0 \cdot 3 = 0 \text{ [ns]}$$

$$\text{wspomnienie-wczytanie} = 0,5 \text{ [ns]}$$

$$h_3 \cdot 16 + 3 = 6 \cdot 16 + 3 = 99_{10}$$

$$10000_{16} = 65536_{10}$$

$$65536 \cdot 99 = 6488064_{10}$$

$$\begin{aligned} \text{zes-iteracji} &= 1 + 0,5 + 1 + 0,5 \\ &\quad + 1 + 0,5 + 7 + 1 + 0,5 = 13 \text{ [ns]} \end{aligned}$$

$$\begin{aligned} \text{zes-pętli} &= 13 \cdot 6488064 = 84344832 \text{ [ns]} = \\ &= \underline{\underline{84344832}} \text{ [ms]} \end{aligned}$$

**Zadanie 5. 2014 termin 1**

5. (Sp) Wymienić z przykładami tryby adresowania architektury x86.

*Było w yzej*

Koło 2016, TERMIN 1 i 2 (bo wiele się powtarza) - to prawdopodobnie będzie na 0 TERMINIE 2017 (jak wy sie z tego rozczytaliscie?)

Organizacja i Architektura Komputerów, Enero 2016, termin 1 - 24.06.2016.

Czas: 40 min. Użyczenie kalkulatorów, zabranione. Do poznania: 10

uwieczniać mi rozstrzyganie niezwykłych niejednoznacznych. Zycie Wom powiedzenia - Piotr Patronik Imię i nazwisko: Stanisław Józefowicz

WING & WILSON  
OZARK CHIEF DIVISION

Działania humanitarnego zapoczątkowane 1 lipca Wspomnianym dniem, zakończyły się

1. (5p) W procesorze o 8-bitowej szynie danych jest dana pamięć podlegająca z-index, w której  $R_1 \times MMR$  jest etykietą adresową. Po trybnie transferu blokowego wyszła  $2^{16-1}$  bajtów. Jaki jest rozmiar pamięci podlegającej?

2. (5p) Jest dana liczba  $X = (1,6 \cdot 10^{-3} + 5 \cdot 10^{-4}) \cdot 10^{-1}$ . Zapisz ją liczbę w postaci pojedynczej przeciążonej zgodnie z normą IEEE 754, a następnie obliczyć jej pierwiastek korzystając z przybliżenia  $\sqrt{1+x} \approx 1+0,5x$  dla  $x \ll 1$ , o której mowało się.

1

Digitized by Google

3. Other Product: *soorten en hoeveelheden voorraad* (voedsel, water, geneesmiddelen, uitrusting, gereedschappen enz.)

4. (Sp) Jest dany fragment kodu pewnego procesora (z systemem z modeliem programowym x86). Napisz rozkaz ruch (będący rozszerzeniem modelu) ladujące rejestr wartościami zmiennej literowej reg rozkładu jednostajnego z przedziału od 0 do wartości z rejestru, odczyt rozkazu z pamięci zajmujące *4c11*<sub>16</sub>, zapis danej do pamięci *4e11*<sub>16</sub>, oraz wykonanie rozkazu *0f84*. Pamięć dla procesora jest przydzielona, dla każdego adresu istnieje odpowiadająca z pamięcią procesora na poziomie fizycznego. Jaki będzie czas wykonania kodu powyższego skryptant sint i end?

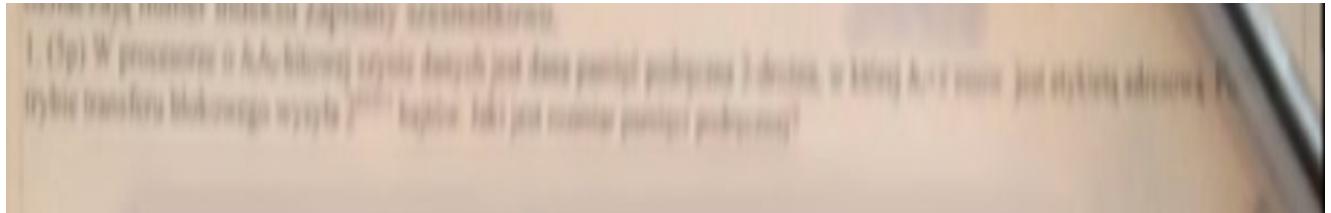
5. Zdjęcie Dostarczane po dniu powrotu zezwolenia programu na założycie gospodarki lokalnej (07.02.0 - przestrzegaj), i o której

#### 6. (10) Wyodrębnij etapy tego procesu w systemie operacyjnym (graf)

# Koło 2016, TERMIN 1 i 2

## ROZWIĄZANIA:

### Zadanie 1. 2016 termin 1 i 2



Obliczenie wielkości pamięci podręcznej (było wyżej podobne) 5pkt // drugi sposób dawał 5/5

W procesorze o h1h0 bitowej szynie danych jest dana 2-drożna pamięć podręczna, w której h3+5 jest etykietą adresową. W trybie transferu blokowego przesyła  $2^{(h1+h2)}$  bajtów. Jaki jest rozmiar pamięci podręcznej.

Analogicznie do poprzednich:

2 1 8 2 4 6

H5 h4 h3 h2 h1 h0

**Szerokość szyny = h1h0 = 46 bitów**

**Drożność = 2**

**Etykieta adresowa = 13 bitów**

**Transfer(blok) =  $2^{(h1+h2)}$  bajtów =  $2^{(4+2)}$  B =  $2^6$ B = 64B**

**Offset = log<sub>2</sub>(transfer) = log<sub>2</sub>(2<sup>6</sup>) = 6**

**Set = szyna - etykieta adresowa - offset = 46 - 13 - 6 = 27**

**Liczba setów =  $2^{(set)} = 2^{27}$**

**Rozmiar pamięci = drożność \* liczba setów \* transfer(blok) =  $2 * 2^{27} * 2^6$ B =**

**$2^{34}$ B =  $2^{24}$ KB =  $2^{14}$ MB jakie kto jednostki woli**

OK

## Zadanie 2. 2016 termin 1 i 2

2. (Op) Jest dana liczba  $X = (1.101)^{10} \times 1.111111111111111$ . Zapisz w formie pojedynczej precyzyji zgodnie z normą IEEE 754, a następnie obliczyc jej pierwiastek korzystajac z przyblizenia (1.000000000000000)^{1/2} = 1.000000000000000.

$X =$

$\sqrt{X} =$

Zapisać liczbę w formacie float (single) oraz obliczyć pierwiastek (korzystając z przybliżenia) i także ją zapisać w tym formacie 5pkt// podobne jak w zerowym terminie, to co ktos tu napisał dawało 5/5 //To jest gdzies zrobione? W którym miejscu? // ta sama zasada co z sześciem, tylko wzorek inny

**Na TERMIN 2 było trzeba obliczyć sześcian zamiast pierwiastka**

## Zadanie 3. 2016 termin 1

3. (Op) Podaj wartości specjalne formatu IEEE 754 pojedynczej precyzyji oraz zakresy wartości zdenormalizowanych

3. (3p) Podać wartości specjalne formatu IEEE 754 pojedynczej precyzyji oraz zakresy wartości zdenormalizowanych.

w. specjalna	znak	bity wykładnika	bity mantysy	wartości wykładnika	Zakres wartości
I. zdenormalizo wana	obojętnie	0000 0000	co najmniej 1 jedynka, <b>brak</b> <b>ukrytej</b> <b>jedynki</b>	-126	$+2^{-126}*0,$ 1..1 $+2^{-149}*0,$ 0..1
$+\infty$	$+$ / -	1111 1111	same 0	---	0
$+\text{zero}$	$+$ / -	0000 0000	same 0	---	0
NaN	obojętnie	1111 1111	co najmniej 1 jedynka	---	---

**Zakres liczb zdenormalizowanych:**

**Dodatnie:**

**Największa:  $2^{(-126)} * 1.11111111111111111111111$**

**Najmniejsza:  $2^{(-126)} * 0.0000000000000000000000001 = 2^{(-149)}$**

//-126 to najmniejszy wykładnik (również w zdenormalizowanej). Różnicą jest brak ukrytej jedynki - dlatego jest mnożenie \*0,cośtam.

//a przypadkiem w zdenormalizowanej wykładnik to nie -127? Czy tak nie jest w standardzie?

//-126 na pewno, mimo, że wykładnik wskazuje inaczej (masz o tym w linku na wiki niżej)

/ok dzięki

Ujemne:

Na odwroć

Czyli jak????

Nk potwierdzi!

**DOBRZE, POTWIERDZAM JEST ZAJEBIŚCIE**

### Zadanie 3. 2016 termin 2

Podać wartości specjalne formatu IEEE 754 pojedynczej precyzji oraz zakresy wartości zdenormalizowanych i znormalizowanych. (podobne do zad 3 z terminu 1)

**JEST WYŻEJ**

Wartosci dla znormalizowanych

Dodatnie

Min  $1,0 \times 2^{-126}$

Max  $1,1\dots1$  (23 jedynki po przecinku)  $\times 2^{127}$

Ujemne

Min  $-1,1\dots1 \times 2^{127}$

Max  $-1,0 \times 2^{-126}$

[https://en.wikipedia.org/wiki/Single-precision\\_floating-point\\_format](https://en.wikipedia.org/wiki/Single-precision_floating-point_format)

## Zadanie 4. 2016 termin 1

4. (5p) Jaki dany fragment kodu programu procesor w systemie z modelami programowymi i RR. Napisz rokaz ranci (dysplay rozwojowym modelu) ladujac rejestr startowy zmiennej koncowej reg rozkazu jednolatopiego z przedzialu od 0 do wartosci z rejestr, obliczaj rozkazu z posligni zajmujace 8x16s, zapis danej do portu 4x16s, czas wykonania rozkazu 8ms. Punkt dla procesor jest przydzielony, dla klocka adresu istniaja odwzorcowosc z posligni procesor na pamci fizyczna, jaki typu czas wykonania kodu przypisany styczeni start i end?

```
start:
    mov  $Au*16#3, %ax
    mov  $0x10000, %alit
    add  %alit, %ax
    mov  %ax, %ax

begin:    mov  %ax, %ax
        add  %ax, %ax
        mov  %ax, (%ax)
loop: begin
end:
```

Może ktoś ten kod rozszyfrować i dokładną treść zadania?

Podany kod asemblera, obliczyć czas wykonania **całego kodu** (było wyżej podobne) 5pkt

Zad4 wie ktoś gdzie można znaleźć info do tego zadania z wykładu Patronika?

// przykład z opracowania starczył + wiadomo CAŁY KOD, a nie tylko pętla

### **Zadanie 4. 2016 termin 2**

Podobne do zadania 4 z 1 terminu (zdjęcie wyżej) przy czym stronicowanie, bez pamięci podręcznej i bez bufora TLB, podany czas wykonania i cykl dostępu kontrolera pamięci h3+1.

**Czy ktoś wie o co chodzi z cyklem dostępu kontrolera pamięci? W sumie nie wiem jak to zadanie wygląda patrząc na ten opis.**

## Zadanie 5. 2016 termin 1

Uzasadnić dlaczego w zadaniu zachodzi lokalność czasowa/przestrzenna.

1. Lokalność czasowa oznacza tendencje do powtarzania odwołań realizowanych w niedawnej przeszłości.
2. Lokalność przestrzenna oznacza tendencje do odwołań do obiektów umieszczonych w obszarze adresowym obejmującym obiekty, które były już użyte w programie.

5. Lokalność czasowa - oznacza tendencję do powtarzania odwołań realizowanych w niedawnej przeszłości.

Lokalność przestrzenna - oznacza tendencję do odwołań do obiektów znajdujących się w obszarze adresowym, w którym znajdują się obiekty, którymi zostały już użyte w programie.

### JAK TO POKAZAĆ?

**W petli sa instrukcje ktore sie powtarzaja petla sie wykonuje kilka razy, instrukcje beda sie wykonywac w tej sekwencji wiele razy. Lokalność czasowa – tendencja do powtarzania odwołań, realizowanych w niedawnej przeszłości (realizacja pętli, referencje do tablicy).**

**Lokalność przestrzenna – tendencja do odwołań do obiektów w obszarze adresowym obejmującym obiekty wcześniej użyte w programie ( kolejne rozkazy programu, elementy regularnej struktury danych, tablice translacji adresów obszar roboczy stosu programowego).**

**Odczytujemy kolejne rozkazy programu z adresow obok siebie w pamieci segmentu kodu czy cos takiego. Pamiec procesu jest dosc duza wiec nwm czy mozna do niej sie odwolac w uzasadnieniu bo to juz nie lokalosc:/No to już chyba zależy od rozmiaru cache, kolego! + jest kilka poziomów cache (rozmiary na potrzeby zadania mogą być abstrakcyjne, tak jak same zadania)**

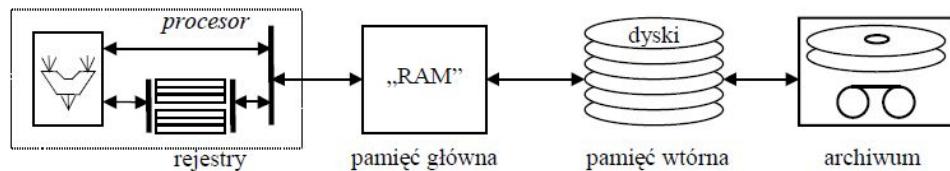
## Zadanie 5. 2016 termin 2

Wymienić elementy hierarchii pamięci z kluczowymi parametrami. (chciał jakis obrazek do tego) // i to nizej bedzie ok?

//to nie będzie to? Tak, będzie

// ten czas, pojemność też mam wymieniać? Serio? <- jeżeli uznać je za kluczowe parametry, to tak

// mówił chyba na wykładzie że to kluczowe parametry



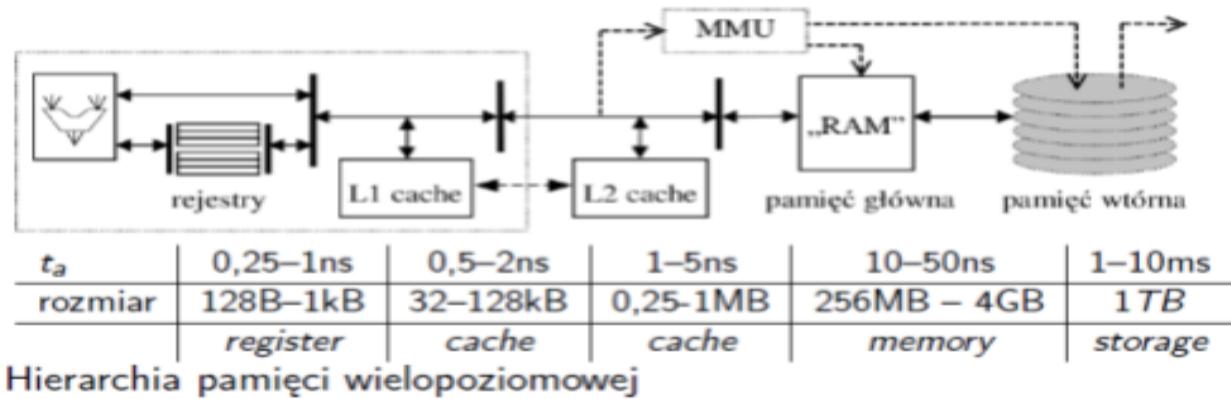
czas dostępu	0,25–1 ns	5–20 ns	1–10 ms	—
pojemność	512B–4 kB	1GB–16GB	>400 GB	>>500 GB
pobór mocy	duży	mały	bardzo mały	bardzo mały
	<i>register</i>	<i>memory</i>	<i>storage</i>	<i>archive</i>

IMO brakuje tutaj cache pomiędzy rejestrami a RAM. Wydaje mi się, że PePe zwracał na to uwagę na wykładzie. Ktoś potwierdzi?

Ja potwierdzę.

<https://drive.google.com/drive/folders/oBxrgd6x3m9NiZUIxQ1JORmhWkE?tid=oBxrgd6x3m9NiUkZQXy1aSDL5YUU> slajd 14 ale który wykład? - Wykład 12

## Pamięć wielopoziomowa



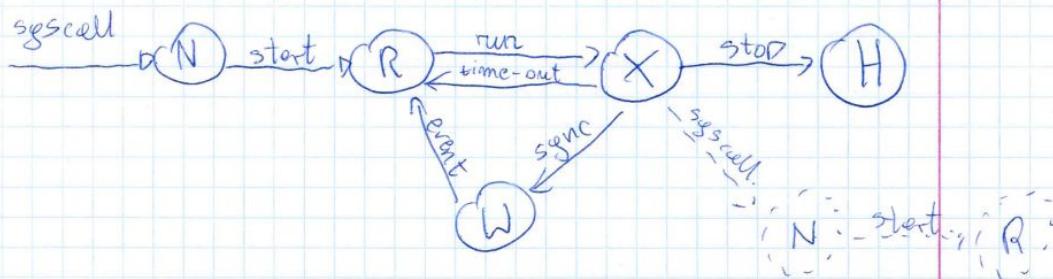
Tu jest wielopoziomowa czy to nie ma znaczenia?

### Zadanie 6. 2016 termin 1

6. (4p) Przedstawić cykl życia procesu w systemie operacyjnym (graf).

Przedstawić cykl życia procesu w systemie operacyjnym (graf).

+ life cycle procesu w systemie operacyjnym



syscall - numer identyfikatora i priorytetu,  
definicje środowiska

start - wpis do kolejki

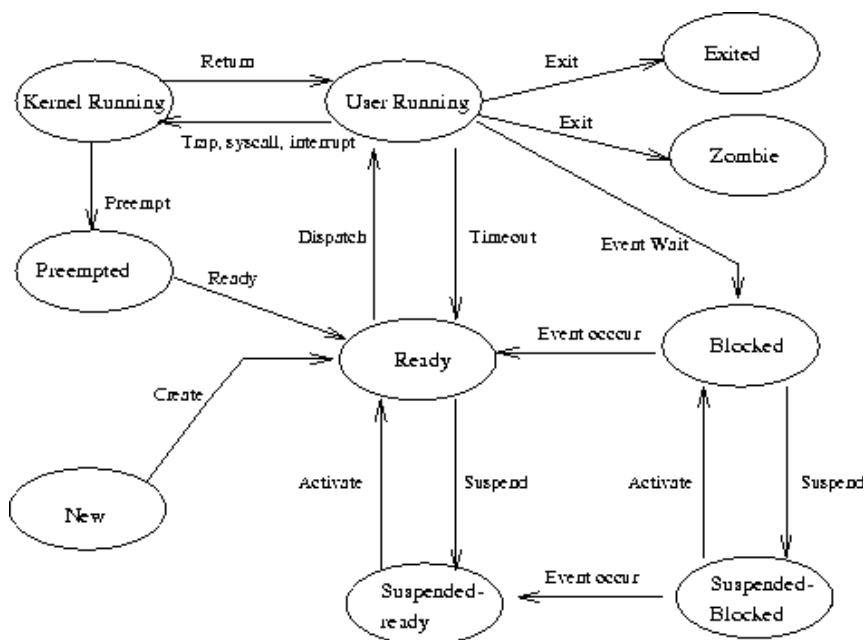
run - otwarcie kontekstu, przeniesienie sterowania,  
usunięcie z kolejki

time-out - wstrzymanie, przeniesienie kontekstu,  
wpis do kolejki

sync - wstrzymanie, przeniesienie kontekstu,  
uspoczenie

event - wznowienie, wpis do kolejki ???

stop - likwidacja środowiska



(<https://www.cim.mcgill.ca/~franco/OpSys-304-427/lecture-notes/node16.html> i bardziej ogólnie:  
<https://www.cim.mcgill.ca/~franco/OpSys-304-427/lecture-notes/node5.html>)

## Zadanie 6. 2016 termin 2

Różnice między RISC i CISC

### WYKŁAD 4

CISC (Complex Instruction Set Computers):

- duża złożoność rozkazów
- rozkazy potrzebują wielu cykli
- złożone (z innych rozkazów) specjalne rozkazy
- dużo trybów adresowania
- do pamięci można odwołać się bezpośrednio z wielu rozkazów
- niższa częstotliwość taktowania niż RISC
- powolny dekoder rozkazów
- sprzętowa Java :-)
- x86 jest przedstawicielem tej architektury

RISC (Reduced/Rational Instruction Set Computers):

- zredukowana ilość instrukcji
- zredukowane tryby adresowania
- load/store z pamięci do rejestrów/z rejestrów do pamięci, pozostałe rozkazy operują wyłącznie na wartościach w rejestrach
- więcej rejestrów (możliwość uninięcia niewykorzystanej komunikacji z RAM-em)
- 1 rozkaz=1 cykl (a przynajmniej do tego się dąży)
- przedstawicielem tej architektury jest ARM, PowerPC (dawne Macintoshe)

## Koncepcja RISC

- ▶ RISC – Reduced Instruction Set Computer
  - !redukcja dotyczy różnorodności, a nie liczby instrukcji (racjonalizacja)!
- ▶ Lista rozkazów
  - ▶ rozkazy proste
  - ▶ proste tryby adresowania
  - ▶ specjalne rozkazy komunikacji z pamięcią
  - ▶ ograniczony zakres i krótkie kody stałych
  - ▶ jednolita struktura kodu
- ▶ Organizacja (zasada lokalności, buforowanie informacji)
  - ▶ dużo rejestrów uniwersalnych
  - ▶ buforowanie informacji (kolejka rozkazów, cache)
- ▶ Korzyści
  - ▶ prawie stały czas wykonania tych samych etapów przetwarzania
  - ▶ podobny czas wykonania różnych etapów przetwarzania
  - ▶ łatwa implementacja potoku
  - ▶ możliwełączenie niektórych rozkazów CISC  
(o minimalnym wpływie na μarchitekturę)

## Architektura klasyczna CISC

- ▶ CISC – Complex Instruction Set Computer
- ▶ Lista rozkazów
  - ▶ rozkazy realizujące działania proste i skomplikowane
  - ▶ rozbudowane sposoby (tryby) adresowania
  - ▶ argumenty umieszczone zwykle w pamięci
  - ▶ stałe w dodatkowych słowach kodu
  - ▶ niejednolita struktura – różna liczba słów kodu
- ▶ Organizacja – rozwiązania intuicyjne
  - ▶ akumulator lub niewiele rejestrów uniwersalnych
  - ▶ większość argumentów w pamięci, rejstry specjalizowane
  - ▶ trudne buforowanie i dekodowanie rozkazów (zmienny rozmiar)
- ▶ Skutki
  - ▶ zmienny czas wykonania tych samych etapów przetwarzania
  - ▶ bariera przepustowości pamięci

**Termin 0 2017**

**To samo co termin 0 2016 tylko ostatnie zadanie zmienione.**

### Ochrona procesu

*Zasady ochrony zasobów:*

- zapobieganie (ang. *prevention*) naruszeniu spójności systemu
- reagowanie na ingerencję w mechanizm ochrony
  - wykrywanie (ang. *detection*) błędów i ataków
  - rozpoznawanie i neutralizacja skutków ingerencji
  - unieważnianie (ang. *nulifying*) działań ingerujących w mechanizm ochrony

*Wspomaganie ochrony zasobów na poziomie architektury rzeczywistej:*

- *w przestrzeni kodów* – uniemożliwienie wykonania instrukcji uprzywilejowanych (ang. *privileged*) w procesie użytkownika
- *w przestrzeni operandów* – wykluczenie wykonania w trybie użytkownika operacji używających zastrzeżonych operandów
- *w przestrzeni danych* – kontrolowany dostęp do danych,  
(przypisanie danej znacznika (ang. *tag*) jest sprzeczne z koncepcją pamięci)
- *furtki* (ang. *gate*) – kontrolowany dostęp do procesów uprzywilejowanych

// w 5 zadaniu jest 6 linijek - na slajdach adresowań było 7 - rezygnowaliście z któregoś czy dopisywaliście mimo braku miejsca?

**// 6 to 6, ja tam nie pisałam dodatkowego**

Organizacja i Architektura Komputerów. Egzamin, termin 0, 20.06.2017  
Czas: 40 min. Używanie kalkulatorów: zabronione. Do notatek służy druga strona kartki. Przejrzysty zapis obliczeń ułatwia mi rozstrzyganie przypadków niejednoznacznych. Zyczę Wam powodzenia – Piotr Patronik  
Imię i nazwisko: ..... Numer indeksu: .....  $h_3 h_3 h_2 h_1 h_0$  Punkty: ...../25. Niech  $h_i$  oznaczają cyfry szesnastkowe numeru indeksu.

1. (4p) Jest dany procesor o 4-bitowym słowie rozkazowym i poniższym kodowaniem rozkazów, w którym cza wykonania każdej mikrooperacji wynosi 50ps. Zapisać program w postaci mnemoników. Zobrazować i podać czas wykonania 6 rozkazów (zapisanych szesnastkowo):  $h_3, h_3, 0xFE, h_0, h_1$  dwóch przypadkach: (i) procesor jest w pełni sekwencyjny, (ii) mikrooperacje (F, W), (F, E), (D, W) mogą być wykonane równocześnie w potoku.

Kod	Zapis	Operacja	Mikrooperacje
$ii\ v\ 1$	$id\ \$v,\ @ri$	$ri \leftarrow v$	FDW
$ii\ j\ 0$	$add\ @ri,\ @rj$	$rj \leftarrow ri+rj$	FDREW

2. (5p) Jest dana liczba binarna  $(1+bbbb \cdot 2^{23} + 3 \cdot 2^{23}) \cdot 2^d$  gdzie  $bbbb$  to 4 najmłodsze bity, zaś  $d$  to najmłodsza cyfra dziesiętna numeru indeksu. Zapisać tę liczbę w postaci zgodnej z normą IEEE 754, a następnie obliczyć (i zapisać ją) sześćnastkową reprezentację tej liczby korzystając z przybliżenia  $(1 \pm x)^3 \approx 1 \pm 3x$  dla  $x \ll 1$ .

. (3p) Przedstawić architekturę jednostki wykonawczej procesora o organizacji  $h_1 \% 3 = \dots : 0$  – akumulatora ogólnego, 1 – uniwersalnej/LS, 2 – stosowej (% oznacza operator modulo).

(5p) Jest dany fragment kodu pewnego procesu w systemie z segmentacją stronicowaną, z modelem programów 6. Niech rozkaz rand (będący rozszerzeniem modelu) ładuje rejestr wartością zmiennej losowej wg rozkładu jednostajnego z przedziału od 0 do wartości z rejestrów. (i) Ile takich procesów można równocześnie uaktywnić na stanie efektu migotania, jeżeli rozmiar dostępnej pamięci głównej wynosi 512MB?

```
v $(h0*15+2), %eax
v $0x10000, %ebx
l %ebx
r %eax, %ebx
r $-1, %ecx

(in:
    mov %ebx, %eax
    rand %eax
    mov %eax, (%eax)
p begin
```

-p) Wymienić z przykładami tryby adresowania architektury x86

....., ..... - ..... ADDR = .....  
....., ..... - ..... ADDR = .....

p) Przedstawić:  $h_0 \bmod 2=0$ : zasady ochrony zasobów procesu,  $h_0 \bmod 2=1$ : metody wspomagania ochrony w architekturze

**Jak robiliście zad4 z tym segmentem pamięci??**

**TERMIN 0 2017 - PIĄTEK**

1. takie samo, tylko inny czas wykonywania mikrooperacji
2. - pierwiastek zamiast sześciadanu + napisać dwa zdania na temat użytego zaokrąglenia
3. Przedstawić różnicę między sumatorami CLA i PPA
4. Bardzo podobne, tyle, że bez podanej pamięci głównej i pytanie o minimalną niezbędną pamięć(?) // **minimalny segment pamięci**  
^ KTOŚ COŚ WIE??
5. To samo
6. To samo

;) )

**TERMIN 1 2017 AK2**

1. Podane ilość bitów i liczba operandów sumatora **CSA** oraz A i T sumatora pełnego. Obliczyć A i T całego układu
2. Liczba w IEEE 754, obliczyć pierwiastek wg. wzoru.

Liczba we wzorze była podobna jak z poprzednich lat, jednak przybliżenie wyglądało:

$$(1 + x)^{3/2} \sim 1 + 3/2 x \text{ dla } x \sim 1$$

Trzeba było chyba więc przeprowadzić bardziej skomplikowane operacje, szczególnie warto zauważyć, że jest do dla  $x \sim 1$ , a nie  $x \ll 1$

### 3. Podać wartości specjalne IEEE 754 i zakresy liczb zdenormalizowanych

#### znormalizowane

Min  $1,0 * 2^{-126}$

Max  $1,1\dots1$  (23 jedynki po przecinku)  $* 2^{127}$

#### zdenormalizowane

Min  $2^{(-126)} * 0,0000\ 0000\ 0000\ 0000\ 001 = 2^{(-149)}$

Max  $-1,0 * 2^{-126}$

### Zadanie 3. 2016 termin 2

Podać wartości specjalne formatu IEEE 754 pojedynczej precyzji oraz zakresy wartości zdenormalizowanych i znormalizowanych. (podobne do zad 3 z terminu 1)

#### Wartosci dla znormalizowanych

##### Dodatnie

**Min  $1,0 * 2^{-126}$**

Max  $1,1\dots1$  (23 jedynki po przecinku)  $* 2^{127}$

##### Ujemne

Min  $-1,1\dots1 * 2^{127}$

Max  $-1,0 * 2^{-126}$

[https://en.wikipedia.org/wiki/Single-precision\\_floating-point\\_format](https://en.wikipedia.org/wiki/Single-precision_floating-point_format)

### Zadanie 3. 2016 termin 1

Podać wartości specjalne formatu IEEE 754 pojedynczej precyzji oraz zakresy wartości zdenormalizowanych.

w. specjalna	znak	bitы wykładnika	bitы mantysy	wartości wykładnika	Zakres wartości
I. zdenormalizowana	obojętnie	0000 0000	co najmniej 1 jedynka, <b>brak ukrytej jedynki</b>	-126	$+2^{-126}*0, 1..1$ $+2^{-149}*0, 0..1$
+- inf	+ / -	1111 1111	same 0	---	0
+- zero	+ / -	0000 0000	same 0	---	0
NaN	obojętnie	1111 1111	co najmniej 1 jedynka	---	---

**Zakres liczb zdenormalizowanych:**

**Dodatnie:**

Największa:  $2^{(-126)} * 0,1111\ 1111\ 1111\ 1111\ 1111\ 111$

Najmniejsza:  $2^{(-126)} * 0,0000\ 0000\ 0000\ 0000\ 0000\ 001 = 2^{(-149)}$

**//-126 to najmniejszy wykładnik (również w zdenormalizowanej). Różnicą jest brak ukrytej jedynki - dlatego jest mnożenie \*0,cośtam.**

**//a przypadkiem w zdenormalizowanej wykładnik to nie -127? Czy tak nie jest w standardzie?**

**//-126 na pewno, mimo, że wykładnik wskazuje inaczej (masz o tym w linku na wiki niżej)**

**/ok dzięki**

**Ujemne:**

**Na odwroć**

**Czyli jak????**

**Nk potwierdzi!**

**DOBRZE, POTWIERDZAM JEST ZAJEBIŚCIE**

#### **4. Obliczyć czas wykonywania programu z pętlą i funkcją rand (zadanie jak wyżej).**

Indeks : 218246

```
mov $h0 * 15 + 2, %eax          // eax = 92
mov $0x10000, %ebx              // ebx = 2^16 = 64kb czy kB? // raczej kb a 8 kB/ ok
                                PAMIEC ADRESOWANA BAJTOWO (byte adressed memory)
mul %ebx                      // eax = 65536 * 92= 64kb *92 = 736kB
mov %eax, %ebx                // ebx = 65536 * 92
mov $-1, %ecx                 // ecx = -1

begin:                         // Początek pętli
    mov %ebx, %eax             // eax = ebx = 65536 * 92
    rand %eax                 // eax = losowanie z przedziału <0; 65536 * 92>
    mov %eax, (%eax)           // (eax) = eax
loop begin                      // dekrementacja ecx i skok jeśli ecx != 0
```

**LINK:** [https://en.wikibooks.org/wiki/X86\\_Assembly/Control\\_Flow#Loop\\_Instructions](https://en.wikibooks.org/wiki/X86_Assembly/Control_Flow#Loop_Instructions)

Wg mnie losowanie je

egzaminu podałeś rozwiązańe

Z tego co widzę to tutaj jest więcej zer, dlatego  $2^{20}$ . no ale są cztery zera, nie? XD

To mi sprawdził nie do tego egzaminu :PA co z  $h0*15+2$ ? Nie zauważył chyba, gdzie się podziało mnożenie  $0x10000$  przez to? :C

//czyli jak to obliczyć? Ktoś jeszcze na konsultacjach był i widział że to 512 jest wynikiem?

a CMP sprawdza czy wartość nie jest zerem

a skoro jest -1 to będzie -2, -3, -4 ...  $-2^{32}$  (max ujemna wartość) i az przekręci się do 0 (ostatecznie w takim wypadku coś to zmienia?)

Po prostu pętla wykona się  $2^{32}-1$  razy, tak jak pisal patronus w opisie drugiego terminu z zeszlego roku.

Liczba procesów:

$512 \text{ MB} = 2^{29} [\text{B}]$  //dlaczego tutaj są bajty, a nie bity? //  $\text{MB} = 2^{20}\text{B}$ ,  $512 = 2^9$ ; mnożysz i tyle

#### **BYTE ADRESSED MEMORY - pamięć adresowana bajtowo**

$2^{29} / (2^{16} * 92) = 2^{13} / 92 = \text{floor}(8192 / 92) = 89$  procesów

//I to jest rozwiązanie żeby nie było migotania? Czy nie trzeba robić odstępów między blokami? // a po co robić odstępy między blokami?

//Tak, bez migotania. Z migotaniem mogłoby być uruchomionych więcej procesów np .90. Choć //to też chyba zależy.  $8192/92=89.04347826$ . Bez migotania zaokrąglamy w dół.

//a gdzie w wyniku jest uwzględnione te  $2^{32}-1$  obrotów pętli?-> //to ile razy wykona się pętla nie ma wpływu na rozmiar zbioru roboczego procesu

Na chłopski rozum:

Efekt migotania = efekt szamotania - Suma zbiorów roboczych > rozmiar dostępnej pamięci

Zbiory robocze to zapotrzebowanie procesu na pamięć w stanie wykonywania - co to może być w tym zadaniu? Żadnej danej nie ma podanej. //no ta funkcja rand... ona sypie z przedziału... Czyli to jest  $(2^{16}*92)$  ?? // wwwydaje mi sie ze tak...

#### **INSTRUKCJE UŻYWANE W KODZIE:**

[http://www.cs.put.poznan.pl/adanilecki/inline\\_asm/doda.php#inne-spis](http://www.cs.put.poznan.pl/adanilecki/inline_asm/doda.php#inne-spis)

[https://pl.wikibooks.org/wiki/Asembler\\_x86/Instrukcje/Arytmetyczne](https://pl.wikibooks.org/wiki/Asembler_x86/Instrukcje/Arytmetyczne)

Poczekajcie mam pytanie... Patrząc na opis tego co było rok temu - on jeszcze dał zadanie na drzewo CSA... ktoś pamięta wzory na policzenie opóźnienia tego drzewa? I ile FA trzeba uzyc?

// zaraz to znajdę w materiałach Janusha

// Iguano, tutaj jest częściowa odpowiedź:

[http://lucc.pl/inf/architektura\\_komputerow\\_1/2004\\_-\\_szybkie\\_sumatory\\_dwuargumentowe.pdf](http://lucc.pl/inf/architektura_komputerow_1/2004_-_szybkie_sumatory_dwuargumentowe.pdf)

// na stronie 12, charakterystyka AT (może coś pomoże xD)

$$h_0 \cdot 15 + 2 = 5 \cdot 15 + 2 = 77_{10}$$

$$10000_{16} = 16^4 = 2^{16}$$

rand losuje z:  $[0, 77 \cdot 2^{16}]$

$$\text{pamięć} = 512[\text{MB}] = 2^9[\text{MB}] = 2^9 \cdot 2^{20} [\text{B}] = 2^{29} [\text{B}]$$

$$\text{karta procesor} = \left\lfloor \frac{2^{29}}{77 \cdot 2^{16}} \right\rfloor = \left\lfloor \frac{2^{13}}{77} \right\rfloor = \left\lfloor \frac{8192}{77} \right\rfloor = 106$$

## 5. Dlaczego w powyższym zadaniu występuje lokalność czasowa / przestrzenna (zadanie jak wyżej).

### JAK TO POKAZAĆ?

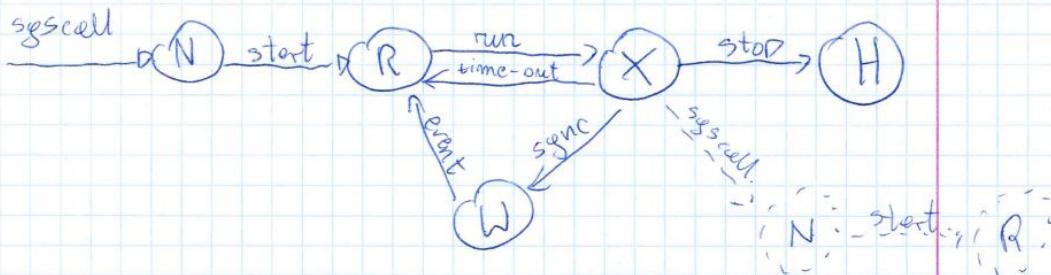
W petli sa instrukcje ktore sie powtarzaja petla sie wykonuje kilka razy, instrukcje beda sie wykonywac w tej sekwencji wiele razy. Lokalność czasowa – tendencja do powtarzania odwołań, realizowanych w niedawnej przeszłości (realizacja pętli, referencje do tablicy).

Lokalność przestrzenna – tendencja do odwołań do obiektów w obszarze adresowym obejmującym obiekty wcześniejsiej uzyte w programie (kolejne rozkazy programu, elementy regularnej struktury danych, tablice translacji adresów obszar roboczy stosu programowego).

Odczytujemy kolejne rozkazy programu z adresow obok siebie w pamieci segmentu kodu czy cos takiego. Pamiec procesu jest dosc duza wiec nwm czy mozna do niej sie odwolac w uzasadnieniu bo to juz nie lokalosc:/No to już chyba zależy od rozmiaru cache, kolego! + jest kilka poziomów cache (rozmiary na potrzeby zadania mogą być abstrakcyjne, tak jak same zadania)

## 6. Przedstawić cykl życia procesu w systemie operacyjnym (graf). (zadanie jak wyżej).

+ Graf życia procesu w systemie operacyjnym



syscall - nadanie identyfikatora i priorytetu,  
definicje środowiska

start - wpis do kolejki

run - odtworzenie kontekstu, przechodzenie sterowania,  
usunięcie z kolejki

time-out - wstrzymanie, przechodzenie kontekstu,  
wpis do kolejki

sync - wstrzymanie, przechodzenie kontekstu,  
uspicie

event - wznowienie, wpis do kolejki ???

stop - likwidacja środowiska

## OiAK 1 termin 29 VI 2017

1 zadanie

2 zadanie zadanie z ieee 754. Należało zrobić pierwiastek liczby. Był chyba wzór  $(1 - x)^{(3/2)}$  w przybliżeniu  $(1 - 3/2x)$

3 zadanie Podać wartości specjalne IEEE 754 i (zakresy ?) coś jeszcze

4 zadanie Obliczyć czas wykonywania programu z pętlą i funkcją rand . w programie było add

5 zadanie lokalność czasowa/ przestrzenna do powyższego

6 zadanie cykl życia procesu

