Java на Эльбрусе

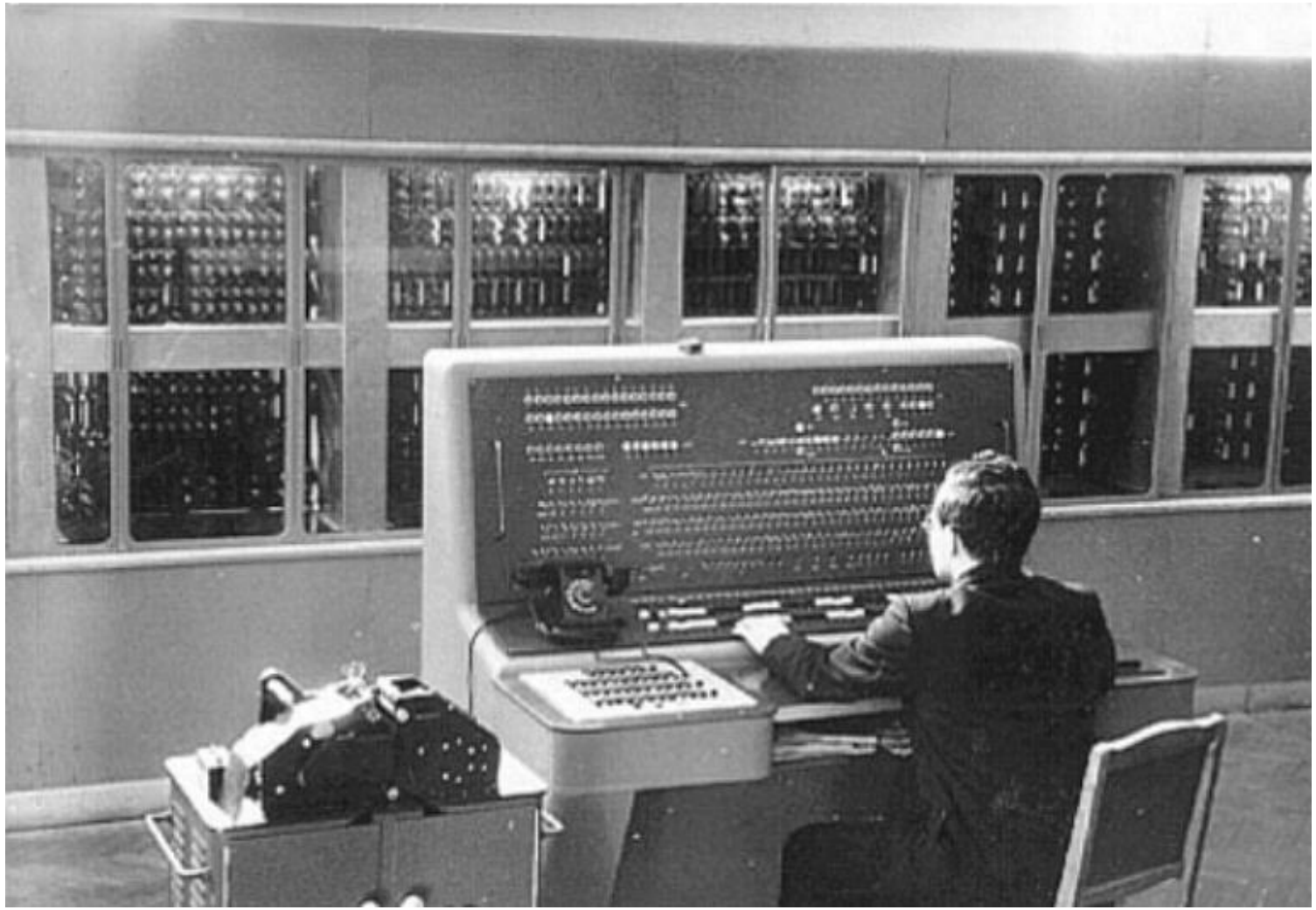
Артемьев Роман

Новосибирский центр информационных технологий

«УНИПРО»

JPoint Student day, 24 апреля 2016

ЭЛЬБРУС*/&?+=?!!!



Эльбрус?


JPoint
ГЛАВНАЯ ПРОГРАММА ДОКЛАДЫ СПОНСОРЫ РЕГИСТРАЦИЯ ОРГАНИЗАЦИЯМ КОНТАКТЫ
JPoint в соцсетях
Спикеры JPoint 2016
JPoint Student Day
специально для студентов
Москва
24 Апреля 2016
Deutsche Bank
МЦСТ
МЦСТ ЭЛЬБРУС

Зачем Эльбрус?

- Импортозамещение
- Безопасность

МЦСТ
ЭЛЬБРУС

Зачем Java на Эльбрусе?

- Java – промышленный стандарт
- Поддержка JVM-based языков
- Шаг на пути к поддержке JavaScript, C#

О чем доклад?

- Про Эльбрус ☺
- Про историю портирования OpenJDK на Эльбрус
- Про компиляторы
- Про Java-runtime
- Про проблемы в работе Java приложений на Эльбрусе
- Про способы увеличения скорости Java на Эльбруса

Кто мы такие?

- Унипро:
  - Участвовали в разработке платформы Java с 1996 г.
  - Разработчик Java Compatibility Kit
  - «Реинкарнатор» Apache Harmony

Отличия архитектур

Эльбрус

x86-64

Отличия архитектур

Эльбрус

- VLIW
- Много регистров (192+32)
- Явная спекулятивность

x86-64

- Суперскалярный
- Мало регистров (16+16+8)
- Неявная спекулятивность

Спекулятивность

```
if (foo != NULL) {
    int a = foo->a + 8;
    float b = foo->b;
    bar(a, b);
}
```

```
int a = foo->a + 8; //spec
float b = foo->b;   //spec
if (foo != NULL) {
    bar(a, b);
}
```

Отличия архитектур

Эльбрус

- VLIW
- Много регистров (192+32)
- Явная спекулятивность
- Условное исполнение

x86-64

- Суперскалярный
- Мало регистров (16+16+8)
- Неявная спекулятивность
- cmov

Условное исполнение

```
if (foo < 5) {
    a += 10;
    ptr->b = 12;
    bar(b, d);
} else {
    ptr->f2 = NULL;
}
```

```
bool pred = foo < 5;
a += 10 ? pred;
ptr->b = 12 ? pred;
bar(b, d) ? pred;
ptr->f2 = NULL ? ~pred;
```

Отличия архитектур

Эльбрус

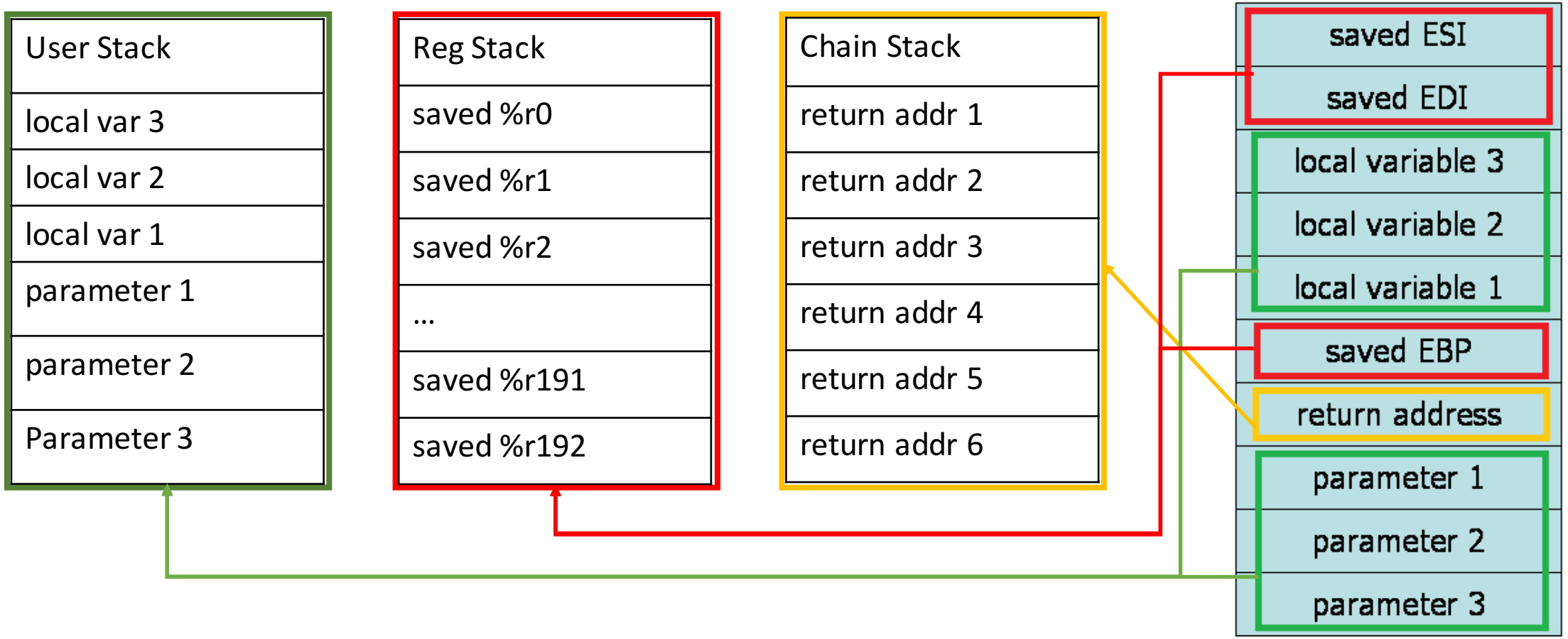
- VLIW
- Много регистров (192+32)
- Явная спекулятивность
- Условное исполнение
- 3 аппаратных стека (2 защищены)

x86-64

- Суперскалярный
- Мало регистров (16+16+8)
- Неявная спекулятивность
- cmov
- 1 аппаратный вседоступный стек

Отличия архитектур

Стеки Эльбруса



Стек x86-64

Отличия архитектур

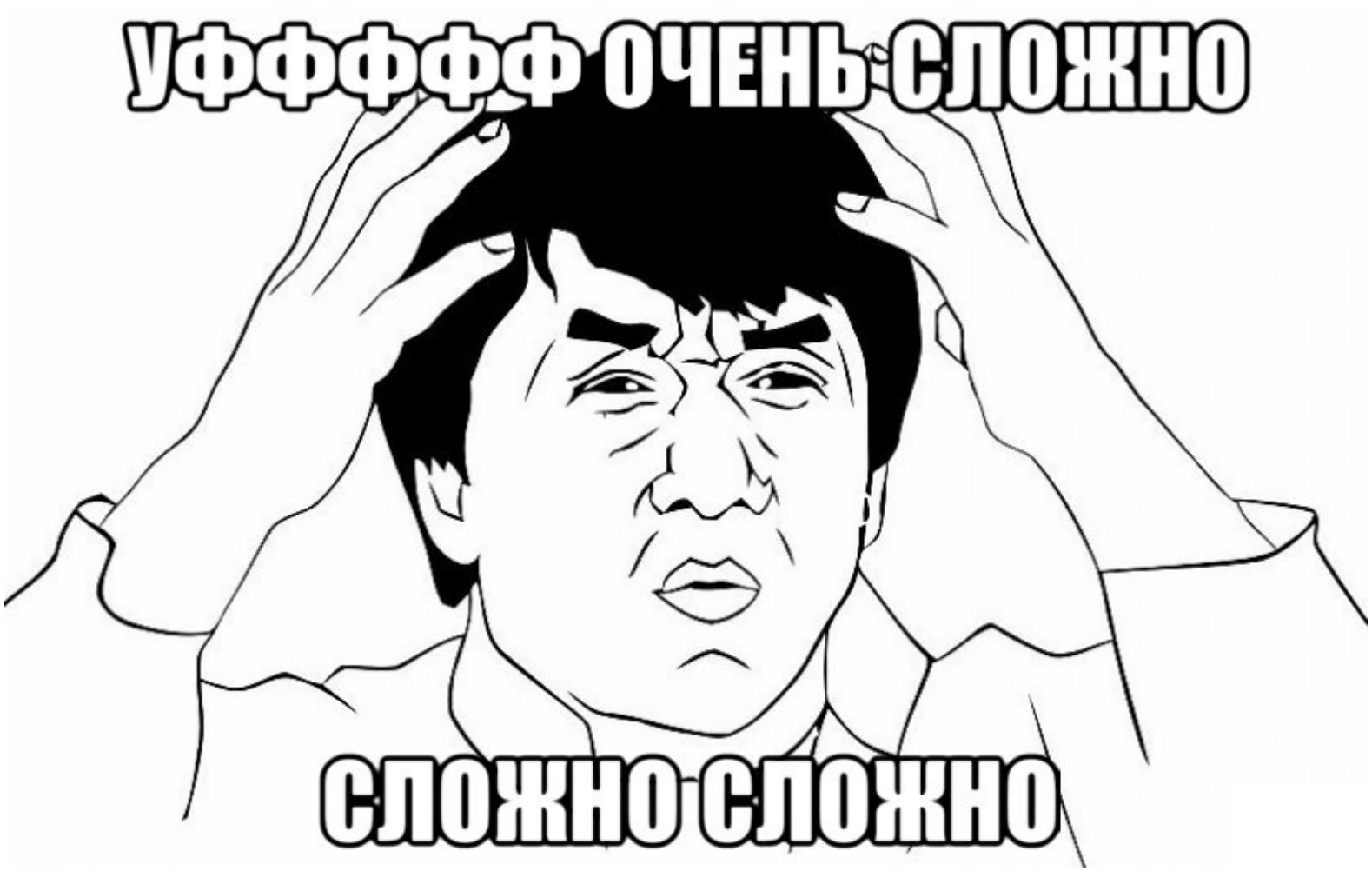
Эльбрус

- VLIW
- Много регистров (192+32)
- Явная спекулятивность
- Условное исполнение
- 3 аппаратных стека
- Нет предсказателя переходов

x86-64

- Суперскалярный
- Мало регистров (16+16+8)
- Неявная спекулятивность
- cmov
- 1 аппаратный стек
- Динамический предсказатель переходов

Свитчи – Зло


УФФФФФФ ОЧЕНЬ СЛОЖНО
СЛОЖНО СЛОЖНО

```
struct Point2D { float x, y; };

float distance(Point2D* a, Point2D* b) {
    return sqrt(
        (a->x - b->x) * (a->x - b->x) +
        (a->y - b->y) * (a->y - b->y));
}
```

```
 1
 2  movss    (%rdi), %xmm1
 3  movss    4(%rdi), %xmm0
 4  subss    (%rsi), %xmm1
 5  subss    4(%rsi), %xmm0
 6  mulss    %xmm1, %xmm1
 7  mulss    %xmm0, %xmm0
 8  addss    %xmm0, %xmm1
 9  sqrtss %xmm1, %xmm0
10  ucomiss %xmm0, %xmm0
11  jp   .L8
12  ret
13
14  .L8:
15  movaps %xmm1, %xmm0
16  pushq    %rax
17  call     sqrtf
18  popq     %rdx
19  ret
```

```
nop 3
setwd       wsz = 0x4
return      %ctpr3
ldw,0       %dr0, 0x0, %r0
ldw,2       %dr0, 0x4, %r1
ldw,3       %dr1, 0x0, %r2
ldw,5       %dr1, 0x4, %r3
```

```
nop 2
fsubs,0     %r0, %r2, %r0
fsubs,3     %r1, %r3, %r1
```

```
nop 4
fmuls,0     %r0, %r0, %r0
fmuls,3     %r1, %r1, %r1
```

```
nop 3
fadds,0     %r0, %r1, %r0
```

```
nop 3
fsqrts,0    %r0, %r0
ct          %ctpr3
```

```
movss    (%rdi), %xmm1
movss    4(%rdi), %xmm0
subss    (%rsi), %xmm1
subss    4(%rsi), %xmm0
mulss    %xmm1, %xmm1
mulss    %xmm0, %xmm0
addss    %xmm0, %xmm1
sqrtss %xmm1, %xmm0
ucomiss %xmm0, %xmm0
jp   .L8
ret

.L8:
movaps %xmm1, %xmm0
pushq    %rax
call     sqrtf
popq     %rdx
ret
```

```
nop 3
setwd       wsz = 0x4
return      %ctpr3
ldw,0       %dr0, 0x0, %r0
ldw,2       %dr0, 0x4, %r1
ldw,3       %dr1, 0x0, %r2
ldw,5       %dr1, 0x4, %r3
```

```
nop 2
fsubs,0     %r0, %r2, %r0
fsubs,3     %r1, %r3, %r1
```

```
nop 4
fmuls,0     %r0, %r0, %r0
fmuls,3     %r1, %r1, %r1
```

```
nop 3
fadds,0     %r0, %r1, %r0
```

```
nop 3
fsqrts,0    %r0, %r0
ct          %ctpr3
```

```
nop 3
setwd       wsz = 0x4
return      %ctpr3
ldw,0       %dr0, 0x0, %r0
ldw,2       %dr0, 0x4, %r1
ldw,3       %dr1, 0x0, %r2
ldw,5       %dr1, 0x4, %r3
```

```
nop 2
fsubs,0     %r0, %r2, %r0
fsubs,3     %r1, %r3, %r1
```

```
nop 4
fmuls,0     %r0, %r0, %r0
fmuls,3     %r1, %r1, %r1
```

```
nop 3
fadds,0     %r0, %r1, %r0
```

```
nop 3
fsqrts,0    %r0, %r0
ct          %ctpr3 ? predX
```

```
nop 3
setwd       wsz = 0x4
return      %ctpr3
ldw,0       %dr0, 0x0, %r0
ldw,2       %dr0, 0x4, %r1
ldw,3       %dr1, 0x0, %r2
ldw,5       %dr1, 0x4, %r3
```

```
nop 2
fsubs,0     %r0, %r2, %r0
fsubs,3     %r1, %r3, %r1
```

```
nop 4
fmuls,0     %r0, %r0, %r0
fmuls,3     %r1, %r1, %r1
```

```
nop 3
fadds,0     %r0, %r1, %r0
```

```
nop 3
fsqrts,0    %r0, %r0
ct          %ctpr3
```

```
nop 3
setwd       wsz = 0x4
return      %ctpr3
ldw,0       %dr0, 0x0, %r0      ; a->x
ldw,2       %dr0, 0x4, %r1      ; a->y
ldw,3       %dr1, 0x0, %r2      ; b->x
ldw,5       %dr1, 0x4, %r3      ; b->y

nop 2
fsubs,0     %r0, %r2, %r0
fsubs,3     %r1, %r3, %r1

nop 4
fmuls,0     %r0, %r0, %r0
fmuls,3     %r1, %r1, %r1

nop 3
fadds,0     %r0, %r1, %r0

nop 3
fsqrts,0    %r0, %r0
ct          %ctpr3
```

```
nop 3
setwd       wsz = 0x4
return      %ctpr3
ldw,0       %dr0, 0x0, %r0
ldw,2       %dr0, 0x4, %r1
ldw,3       %dr1, 0x0, %r2
ldw,5       %dr1, 0x4, %r3

nop 2
fsubs,0     %r0, %r2, %r0       ; (a.x - b.x)
fsubs,3     %r1, %r3, %r1       ; (a.y - b.y)

nop 4
fmuls,0     %r0, %r0, %r0       ; (.x) * (.x)
fmuls,3     %r1, %r1, %r1       ; (.y) * (.y)

nop 3
fadds,0     %r0, %r1, %r0       ; (x*x) + (y*y)

nop 3
fsqrts,0    %r0, %r0            ; sqrt(..)
ct          %ctpr3
```

history | grep elbrus

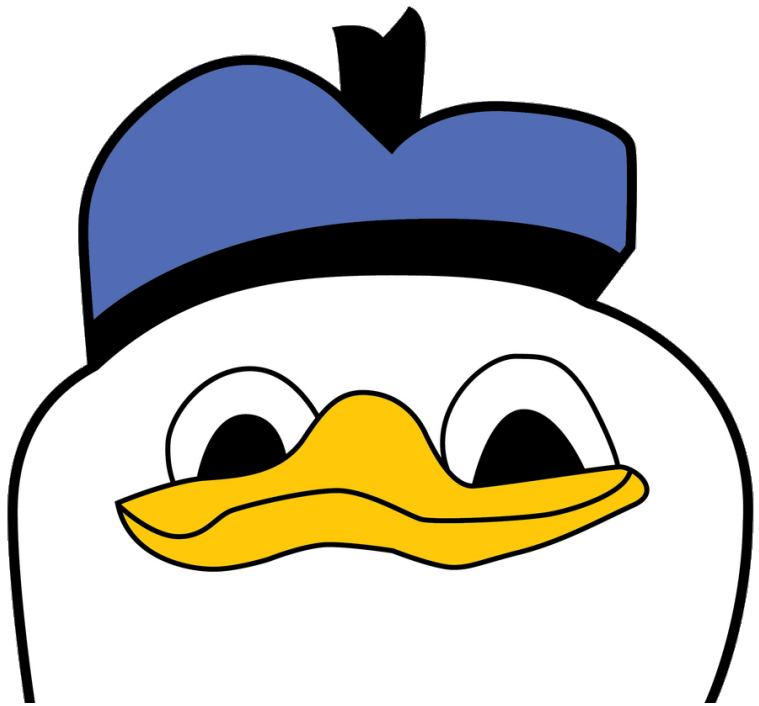
1. OpenJDK + Zero + LLVM
2. VLIW-кодогенератор
3. Client-JIT для Эльбруса
4. Server-JIT для Эльбруса
5. Шаблонный интерпретатор

Version 1.0

- LLVM Backend для Эльбруса
- JIT-компилятор Shark
- Tail call оптимизация CPP-интерпретатора
- На Эльбрусе появилась Java!

Первый блин комом...

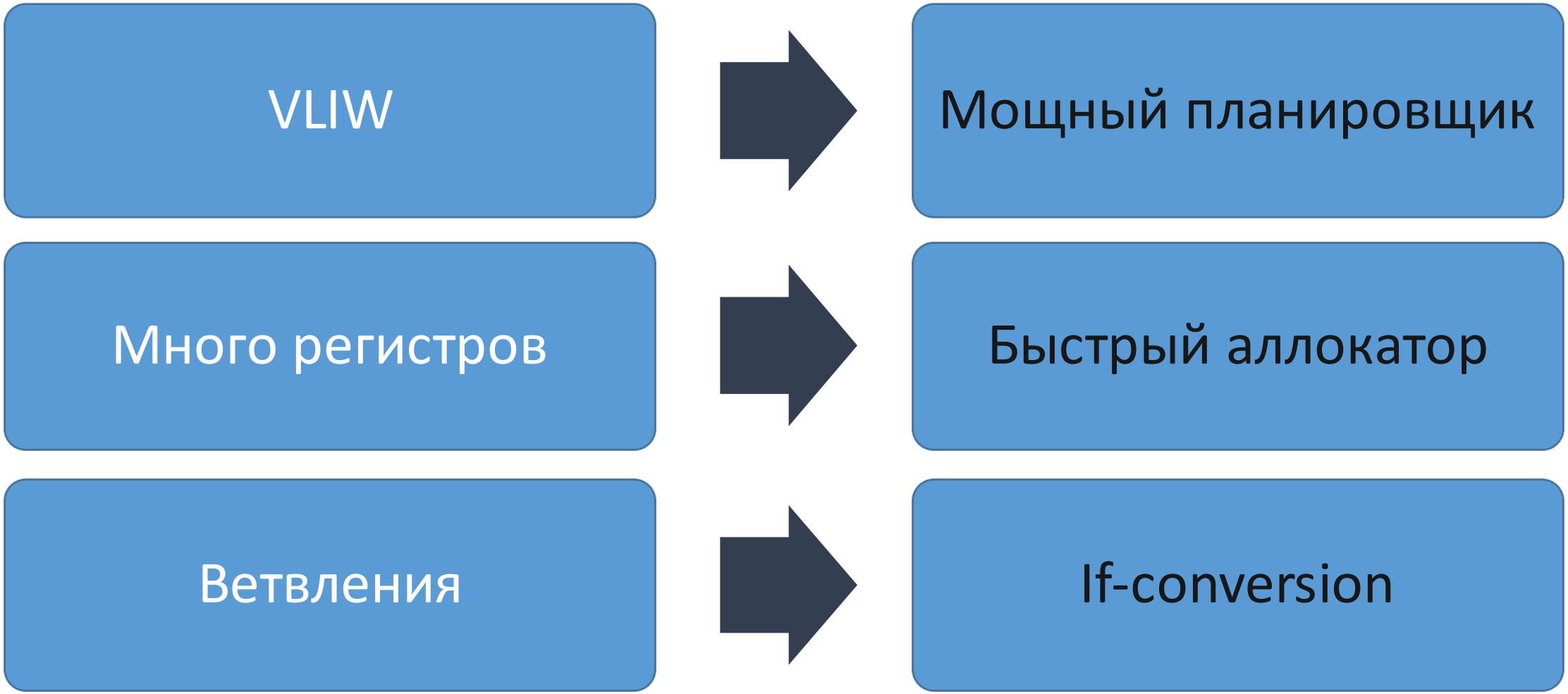
- Много, очень много ветвлений
- Неэффективная сборка мусора
- Явные проверки
- Нет High-level оптимизаций
- Нет Low-level оптимизаций
- И т.д.



Состояние дел

- Дорогие переходы
- Нет Low-level оптимизаций
- Низкая скорость компиляции
- Неэффективная сборка мусора
- Явные Runtime проверки
- Нет High-level оптимизаций
- Медленный интерпретатор

Что же делать?



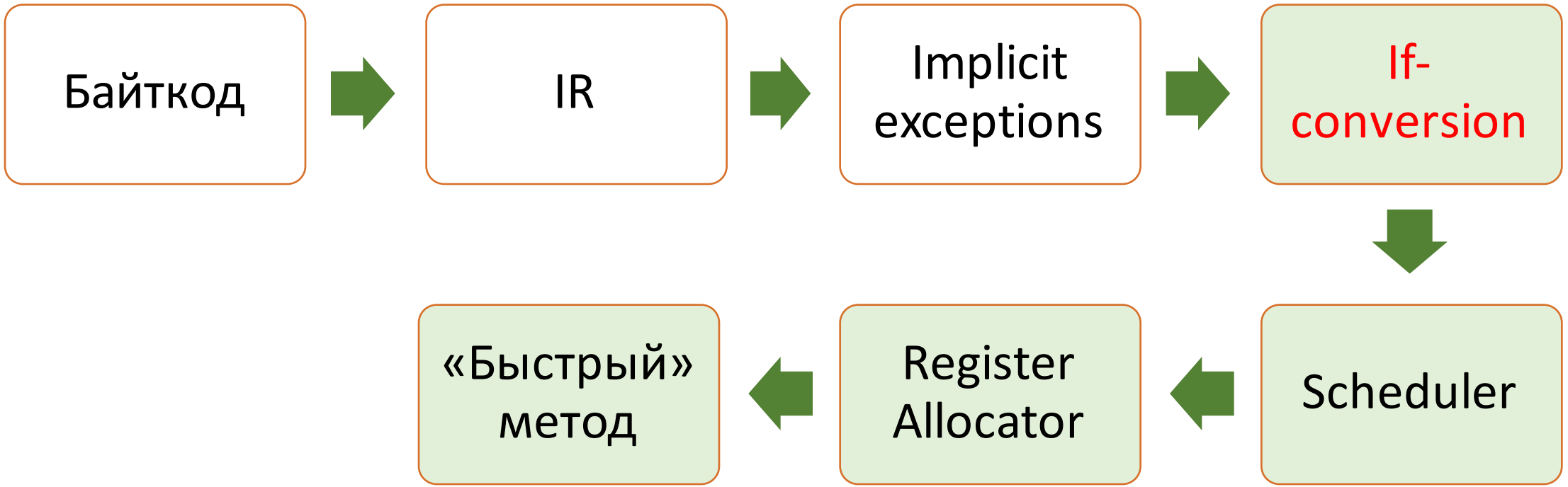
Кодогенератор

- Планировщик – “superblock”
  - Спекулятивность
  - Распределитель управляющих регистров
- Регистры – раскраска графа
- If-conversion – пока нет ☹

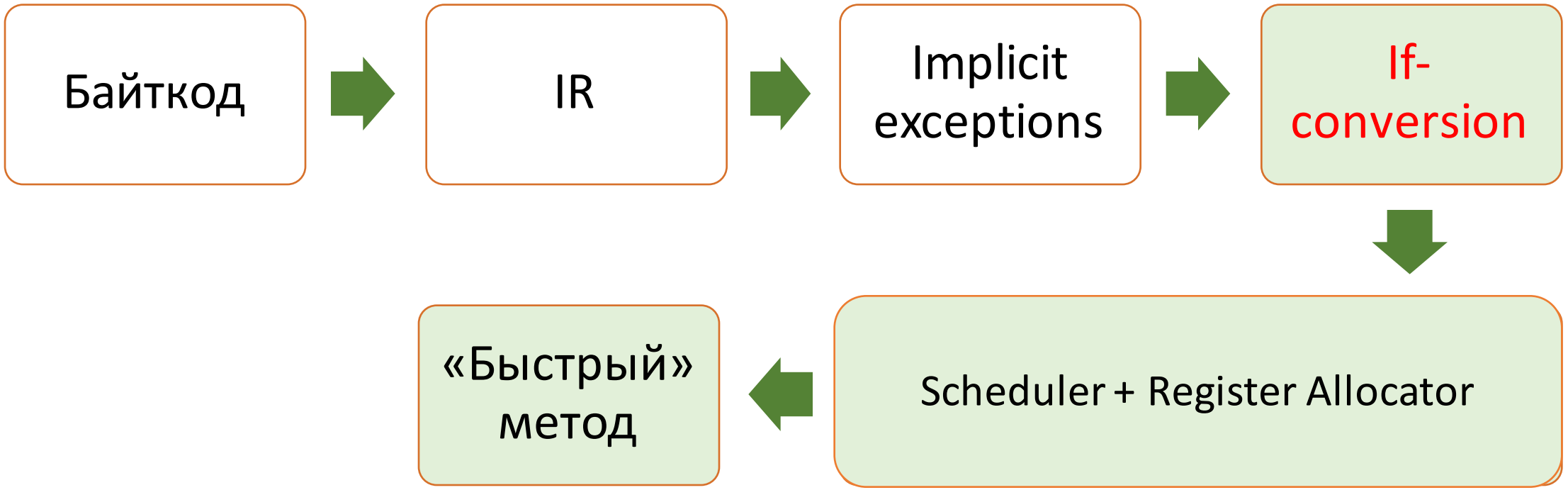
Состояние дел

- Дешевые переходы
- Low-level оптимизации
- Низкая скорость компиляции
- Неэффективная сборка мусора
- Явные Runtime проверки
- Нет High-level оптимизаций
- Медленный интерпретатор

Client-компилятор



Client-компилятор



Состояние дел

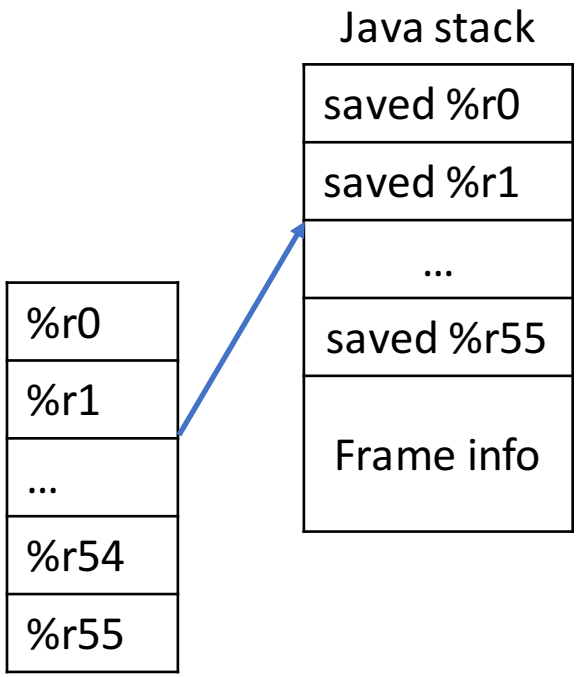
- Дешевые переходы
- Low-level оптимизации
- Высокая скорость компиляции (x10)
- Неэффективная сборка мусора
- Явные Runtime проверки
- Нет High-level оптимизаций
- Медленный интерпретатор

Garbage Collection

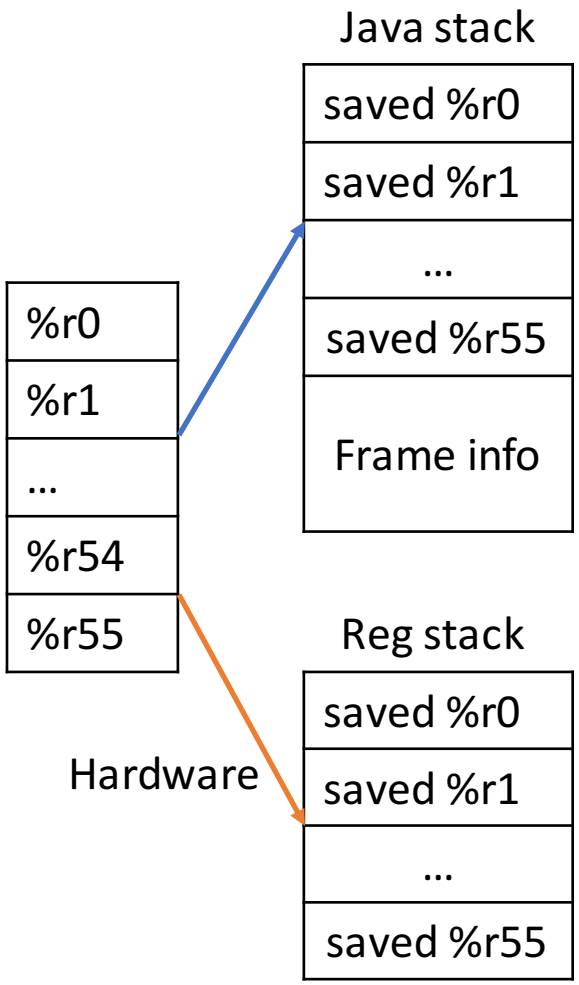
GC: Эльбрус

%r0
%r1
…
%r54
%r55

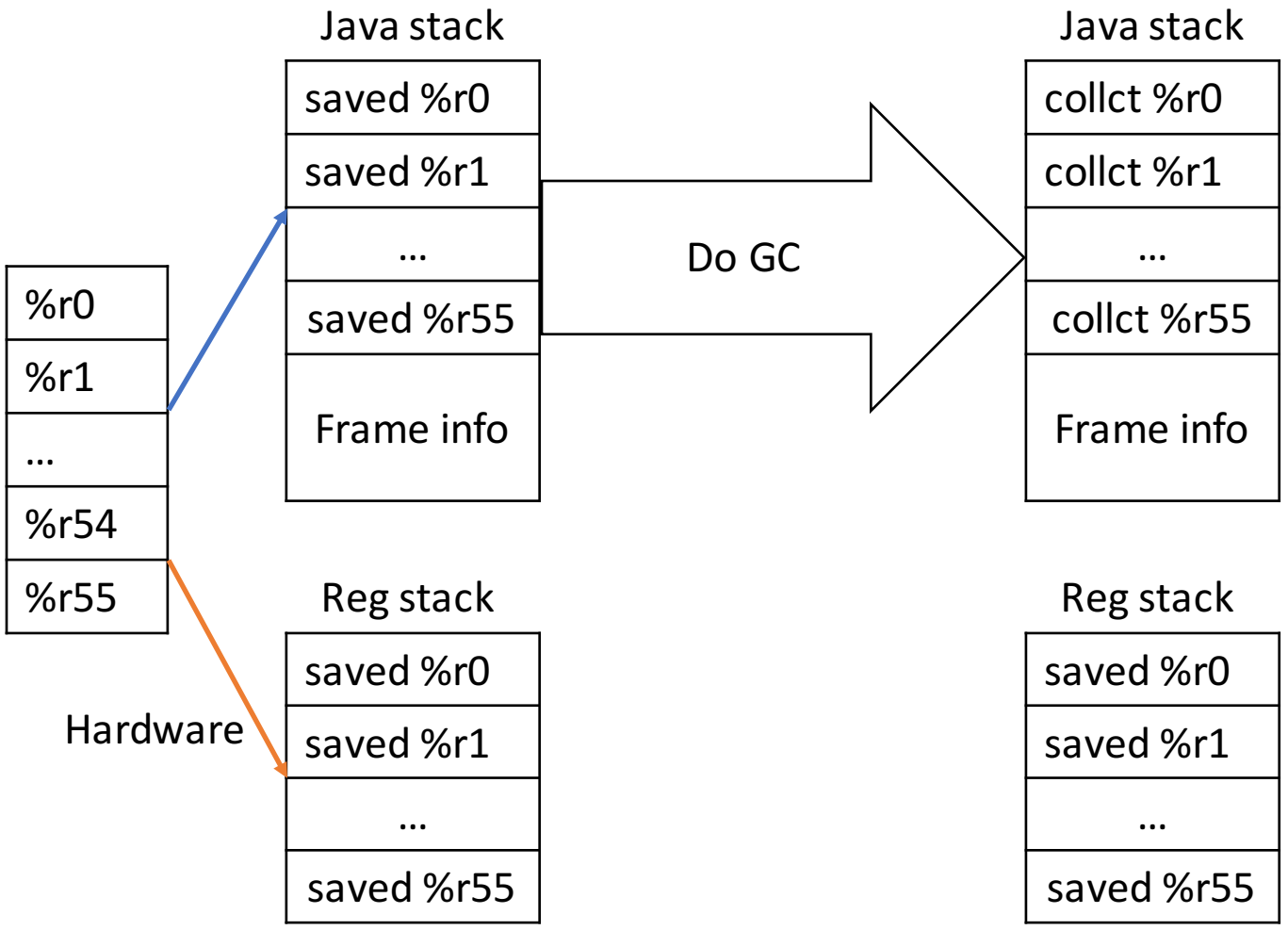
GC: Эльбрус


Java stack
saved %r0
saved %r1
…
saved %r55
Frame info
%r0
%r1
…
%r54
%r55

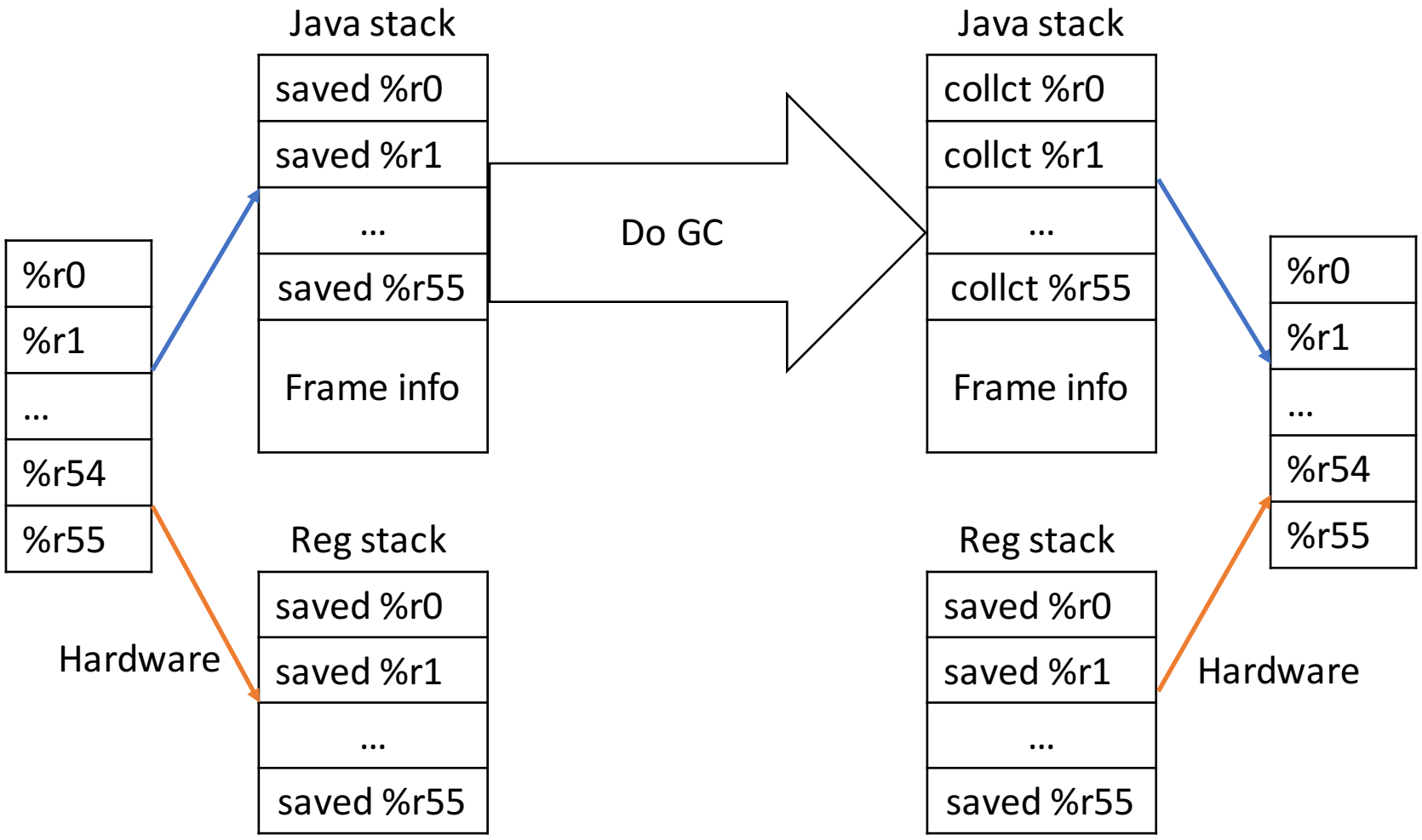
GC: Эльбрус


Java stack
saved %r0
saved %r1
…
saved %r55
Frame info
%r0
%r1
…
%r54
%r55
Reg stack
saved %r0
saved %r1
…
saved %r55
Hardware

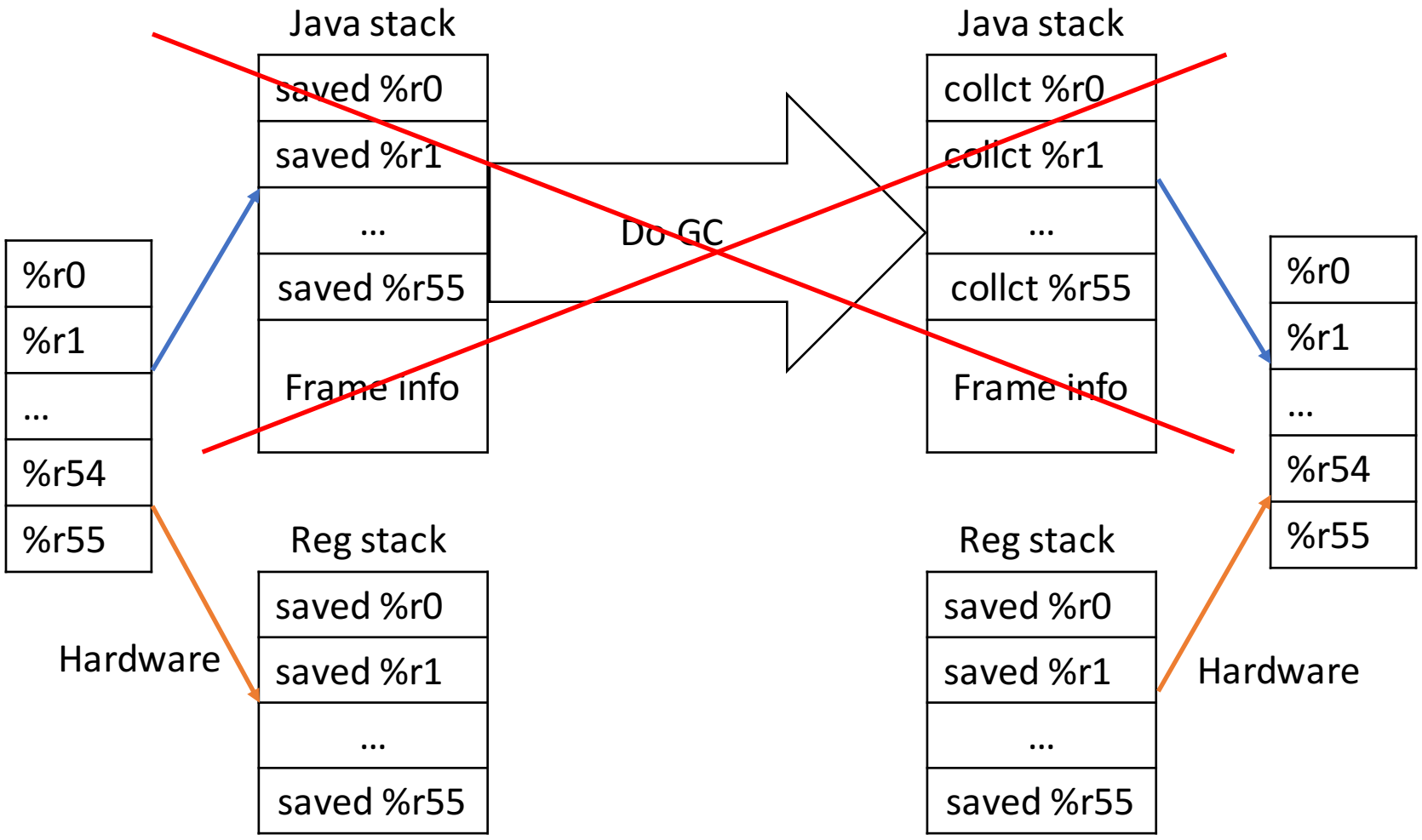
GC: Эльбрус


Java stack
saved %r0
saved %r1
…
saved %r55
Frame info
Do GC
Java stack
collct %r0
collct %r1
…
collct %r55
Frame info
%r0
%r1
…
%r54
%r55
Hardware
Reg stack
saved %r0
saved %r1
…
saved %r55
Reg stack
saved %r0
saved %r1
…
saved %r55

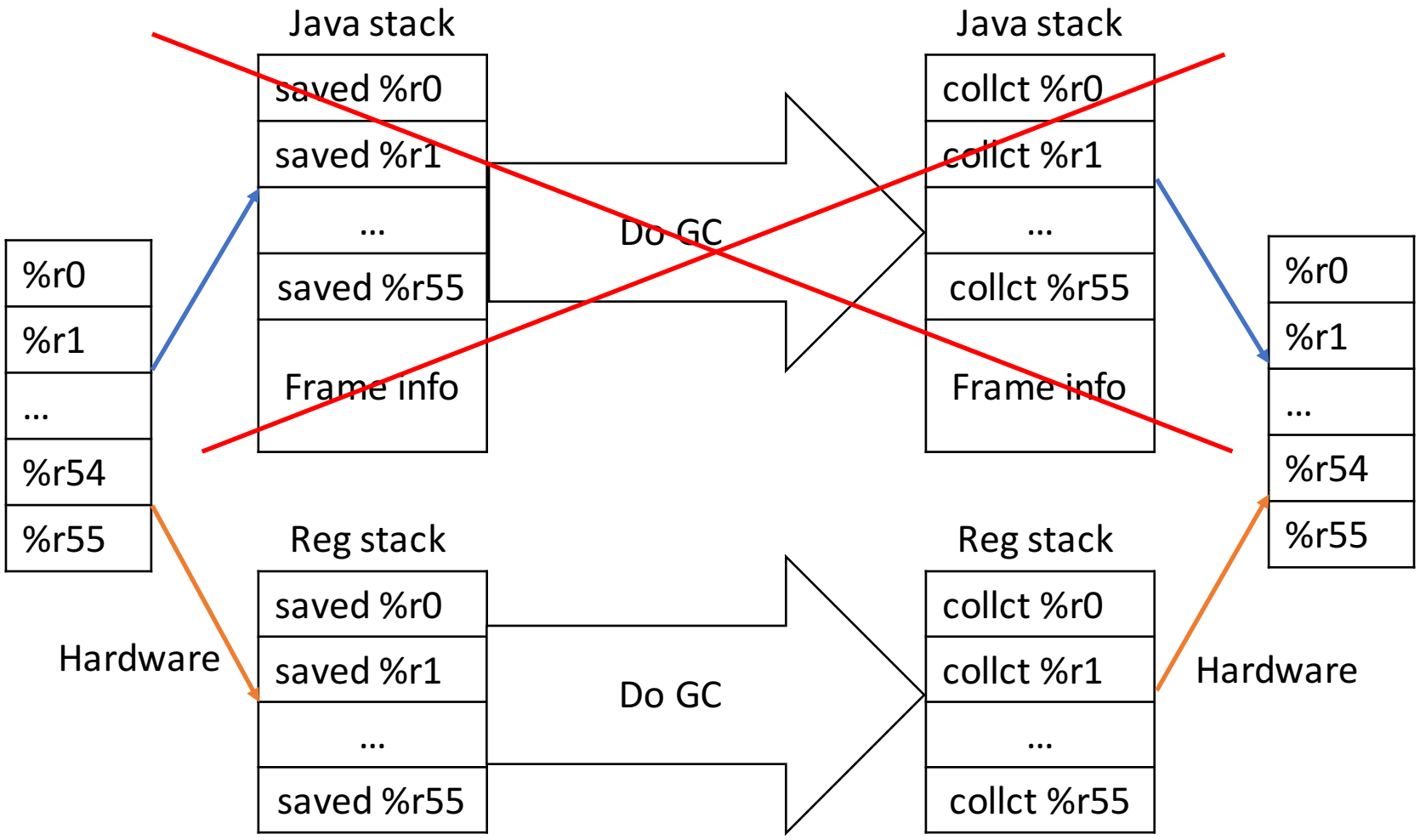
GC: Эльбрус


Java stack
saved %r0
saved %r1
…
saved %r55
Frame info
%r0
%r1
…
%r54
%r55
Hardware
Reg stack
saved %r0
saved %r1
…
saved %r55
Do GC
Java stack
collct %r0
collct %r1
…
collct %r55
Frame info
Reg stack
saved %r0
saved %r1
…
saved %r55
%r0
%r1
…
%r54
%r55
Hardware

GC: Эльбрус


Java stack
saved %r0
saved %r1
…
saved %r55
Frame info
Do GC
Java stack
collct %r0
collct %r1
…
collct %r55
Frame info
%r0
%r1
…
%r54
%r55
Reg stack
saved %r0
saved %r1
…
saved %r55
Hardware
Reg stack
saved %r0
saved %r1
…
saved %r55
Hardware
%r0
%r1
…
%r54
%r55

GC: Эльбрус


Java stack
saved %r0
saved %r1
…
saved %r55
Frame info
Java stack
collct %r0
collct %r1
…
collct %r55
Frame info
Do GC
%r0
%r1
…
%r54
%r55
Reg stack
saved %r0
saved %r1
…
saved %r55
Reg stack
collct %r0
collct %r1
…
collct %r55
Do GC
Hardware
%r0
%r1
…
%r54
%r55
Hardware

Состояние дел

- Дешевые переходы
- Low-level оптимизации
- Высокая скорость компиляции
- Сборка мусора через стек регистров (x2)
- Явные Runtime проверки
- Нет High-level оптимизаций
- Медленный интерпретатор

Runtime проверки

Идея – явное сделать неявным

Runtime проверки

- NPE – ловим SIGSEGV или SIGILL

Null Pointer Exception

load,sm [ptr1 + 0x10], dst «тяжелая» инструкция

Null Pointer Exception

load,sm [ptr1 + 0x10], dst

……

«тяжелая» инструкция

Что-то считаем

Null Pointer Exception

load,sm [ptr1 + 0x10], dst	«тяжелая» инструкция
……	Что-то считаем
add dst, rX, rY	«легкая» инструкция SIGILL -> NPE

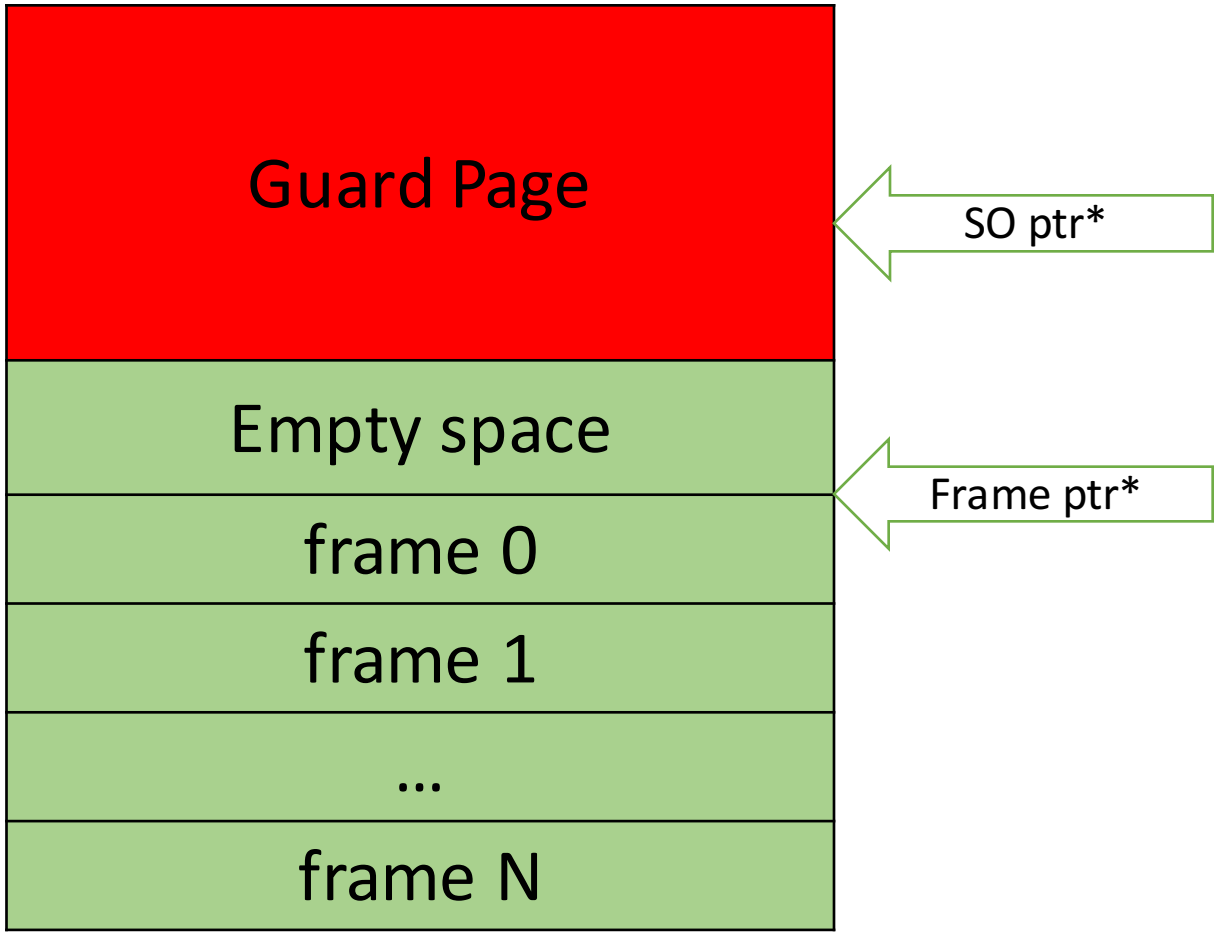
Null Pointer Exception

load,sm [ptr1 + 0x10], dst	«тяжелая» инструкция
……	Что-то считаем
add dst, rX, rY	«легкая» инструкция SIGILL -> NPE
store rY, [ptr2 + 0x20]	SIGSEGV -> NPE

Runtime проверки

- NPE – ловим SIGSEGV или SIGILL
- Div0 – ловим SIGILL
- Stack Overflow – guard page, SIGSEGV

Stack Overflow



Runtime проверки

- NPE – ловим SIGSEGV
- Div0 – ловим SIGILL
- Stack Overflow – guard page, SIGSEGV
- Bounds Check – с помощью условных инструкций провоцируем SIGSEGV при выходе за границу

Bounds Check

```
load arr_ptr, arr_length                    ; SIGSEGV -> NPE
```

Bounds Check

```
load arr_ptr, arr_length                    ; SIGSEGV -> NPE
cmpU, index, arr_length, pred
```

Bounds Check

```
load arr_ptr, arr_length                    ; SIGSEGV -> NPE
cmpU, index, arr_length, pred
mov 0, arr_ptr ? ~pred
mov 0, index ? ~pred
```

Bounds Check

```
load arr_ptr, arr_length                    ; SIGSEGV -> NPE
cmpU, index, arr_length, pred
mov 0, arr_ptr ? ~pred
mov 0, index ? ~pred
load [arr_ptr + index], val                 ; SIGSEGV -> OOB
```

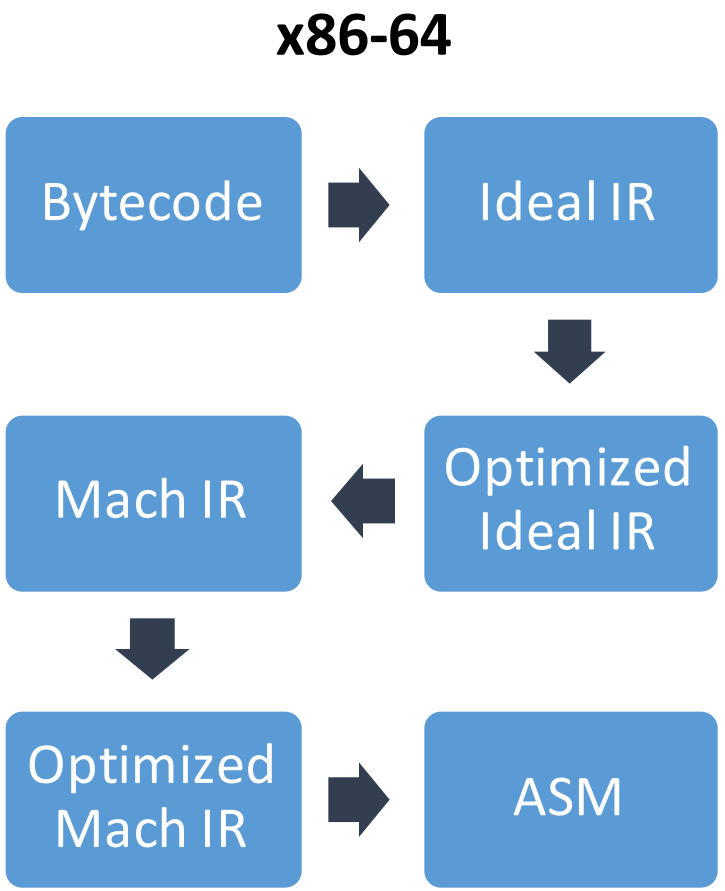
Runtime проверки

- NPE – ловим SIGSEGV
- Div0 – ловим SIGILL
- Stack Overflow – guard page, SIGSEGV
- Bounds Check – с помощью условных инструкций провоцируем SIGSEGV при выходе за границу

Состояние дел

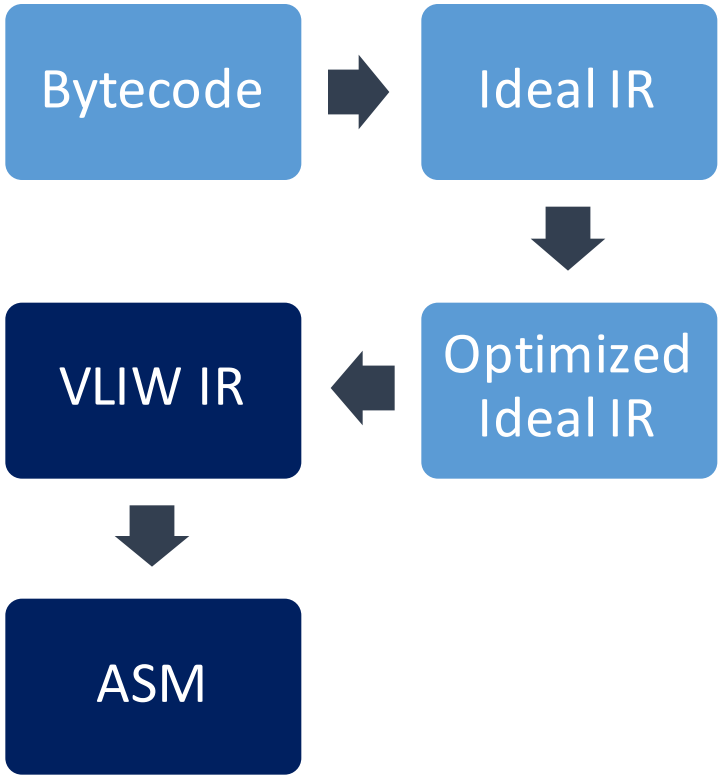
- Дешевые переходы
- Low-level оптимизаций
- Высокая скорость компиляции
- Сборка мусора через стек регистров
- Неявные Runtime проверки (+15%)
- Нет High-level оптимизаций
- Медленный интерпретатор

Opto

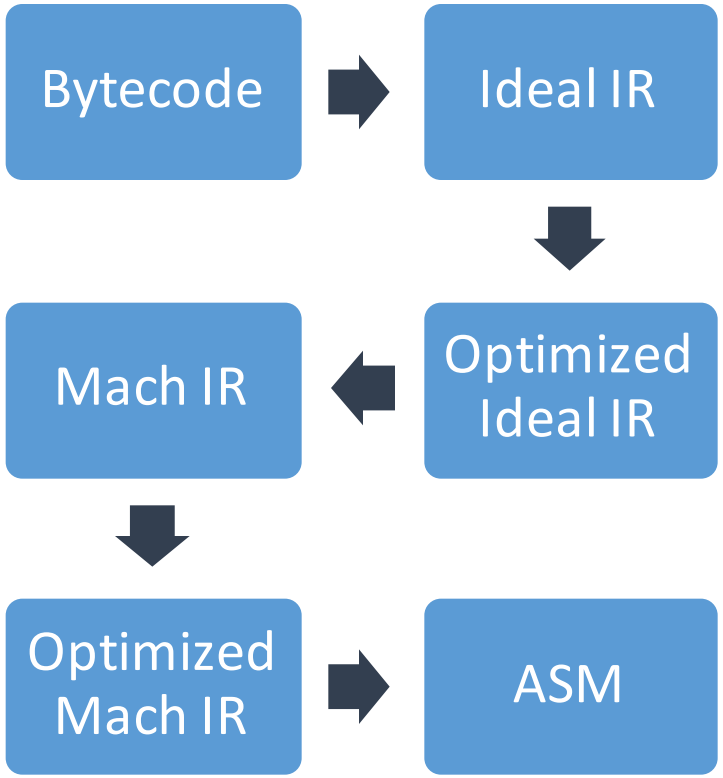

x86-64
Bytecode
Ideal IR
Optimized Ideal IR
Mach IR
Optimized Mach IR
ASM

Opto

Эльбрус



x86-64



Состояние дел

- Дешевые переходы
- Low-level оптимизаций
- Высокая скорость компиляции
- Сборка мусора через стек регистров
- Неявные Runtime проверки
- High-level оптимизаций
- Медленный интерпретатор

Шаблонный интерпретатор

- Влияет на скорость старта приложений
- На x86 быстрее обычного в 10 раз
- Реализация для Эльбруса ведется

Состояние дел

- Дешевые переходы
- Low-level оптимизаций
- Высокая скорость компиляции
- Сборка мусора через стек регистров
- Неявные Runtime проверки
- High-level оптимизаций
- Быстрый интерпретатор

Сравнение режимов компиляции

	Shark	Client	Server	Ускорение Server/Shark
SpecJBB2005, число операций, больше –> лучше				
SpecJBB2005	2172	6261	18378	8.5
SpecJVM98, время в секундах, меньше –> лучше				
compress	76.08	28.56	15.57	4.9
jess	35.26	12.62	5.43	6.5
db	60.78	32.12	19.35	3.14
javac	41.78	18.16	9.74	4.3
mpegaudio	70.58	36.39	10.37	6.8
mtrt	32.06	12.15	1.64	19.5
jack	25.33	15.99	9.92	2.5

Сравнение режимов компиляции

	Shark	Client	Server	Ускорение Server/Shark
SpecJBB2005, число операций, больше –> лучше				
SpecJBB2005	2172	6261	18378	8.5
SpecJVM98, время в секундах, меньше –> лучше				
compress	76.08	28.56	15.57	4.9
jess	35.26	12.62	5.43	6.5
db	60.78	32.12	19.35	3.14
javac	41.78	18.16	9.74	4.3
mpegaudio	70.58	36.39	10.37	6.8
mtrt	32.06	12.15	1.64	19.5
jack	25.33	15.99	9.92	2.5

Сравнение режимов компиляции

	Shark	Client	Server	Ускорение Server/Shark
SpecJVM2008, число операций, больше –> лучше				
compiler	4.3	13.8	25.8	6.0
compress	2.7	11.1	20.8	7.7
crypto	3.1	6.0	20.3	6.5
derby	4.9	17.2	36.9	7.5
mpegaudio	2.3	4.1	12.4	5.4
scimark.large	1.2	1.8	3.8	3.2
scimark.small	4.0	6.8	16.5	4.1
serial	1.5	5.3	12.5	8.3
startup	0.7	1.4	2.3	3.3
sunflow	0.9	4.0	8.2	9.1
xml	5.2	14.1	34.2	6.6
Общий счет	2.3	6.0	13.4	5.8

Тестовый стенд

Эльбрус

- Elbrus 4C
- 800 MHz
- 4 cores
- 65 нм
- 8 Mb L2 Cache

x86-64

- Intel Core2Quad Q9300
- 2.50 GHz
- 4 cores
- 45 нм
- 6 Mb L2 Cache

Сравнение с x86-64

	x86-64	Elbrus	Отношение x86-64/Elbrus
SpecJBB2005, число операций, больше –> лучше			
SpecJBB2005	65682	18378	3.6
SpecJVM98, время в секундах, меньше –> лучше			
compress	2.38	15.57	6.5
jess	0.54	5.43	10.0
db	5.05	19.35	3.8
javac	1.63	9.74	6.0
mpegaudio	1.32	10.37	7.8
mtrt	0.3	1.64	5.5
jack	0.83	9.92	11.9

Сравнение с x86-64

	Intel	Elbrus	Отношение x86-64/Elbrus
SpecJVM2008, число операций, больше –> лучше			
compiler	241.65	25.8	9.4
compress	147.42	20.8	7.1
crypto	152	20.3	7.6
derby	209.91	36.9	5.7
mpegaudio	86.05	12.4	6.9
scimark.large	17.45	3.8	4.6
scimark.small	164.54	16.5	10.0
serial	88.65	12.5	7.1
startup	22.49	2.3	9.8
sunflow	49.49	8.2	6.0
xml	308.56	34.2	9.0
Общий счет	98.59	13.4	7.3

Что дальше?

- Java
  - If-conversion
  - Векторизация
  - Compressed Oops
  - Template Interpreter
  - Concurrent GC
- JavaScript (V8)
- C#

Summary

- Что такое Эльбрус
- Что за Java на Эльбрусе
- Как не надо писать на Java под Эльбрус
- Как устроен JIT-компилятор
- Как портировать Java на «экзотику»
- «Реальную» скорость Эльбруса
- Перспективы Эльбруса

Java на Эльбрус

Артемьев Роман

rartemev@unipro.ru

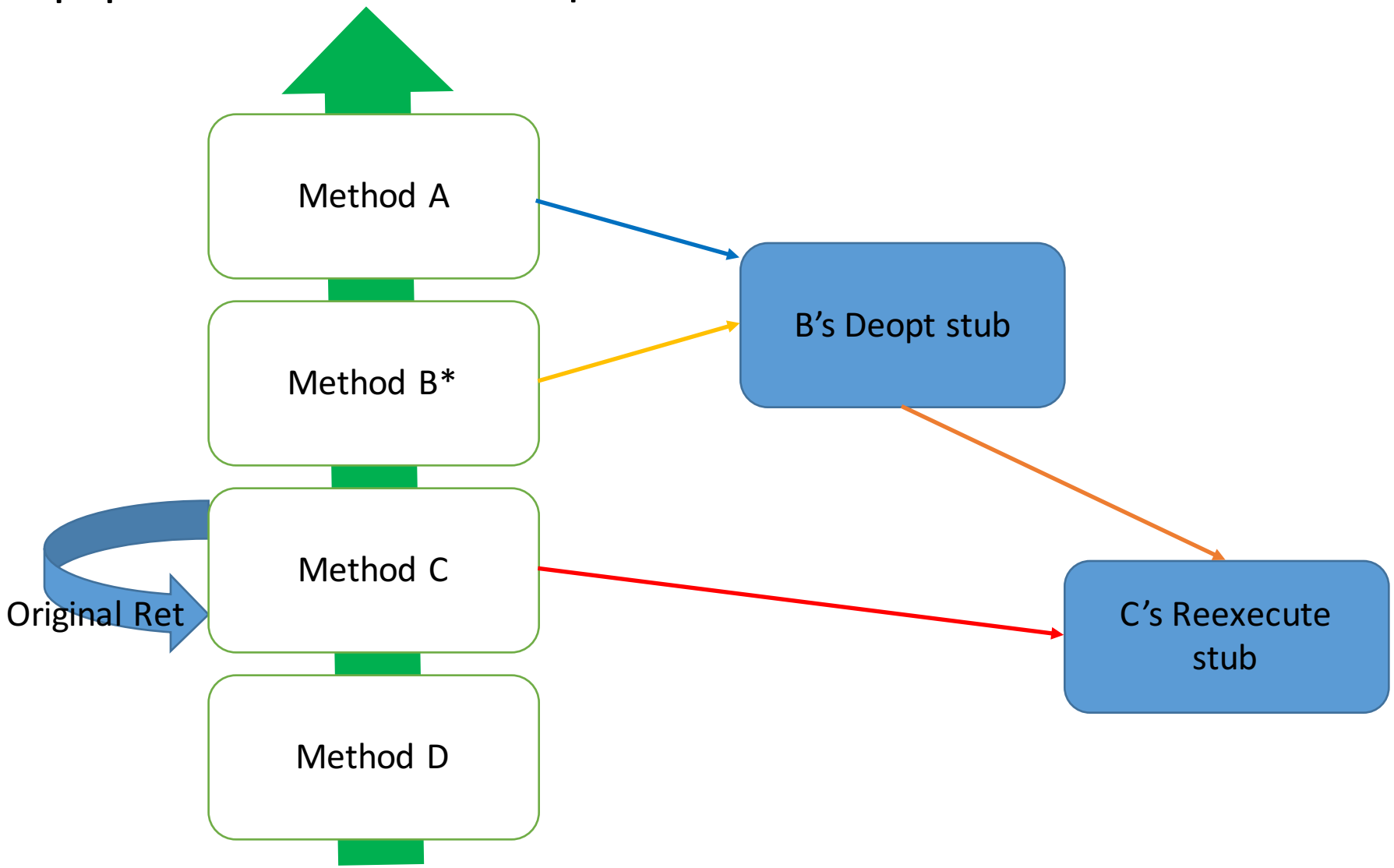
Новосибирский центр информационных технологий

«УНИПРО»

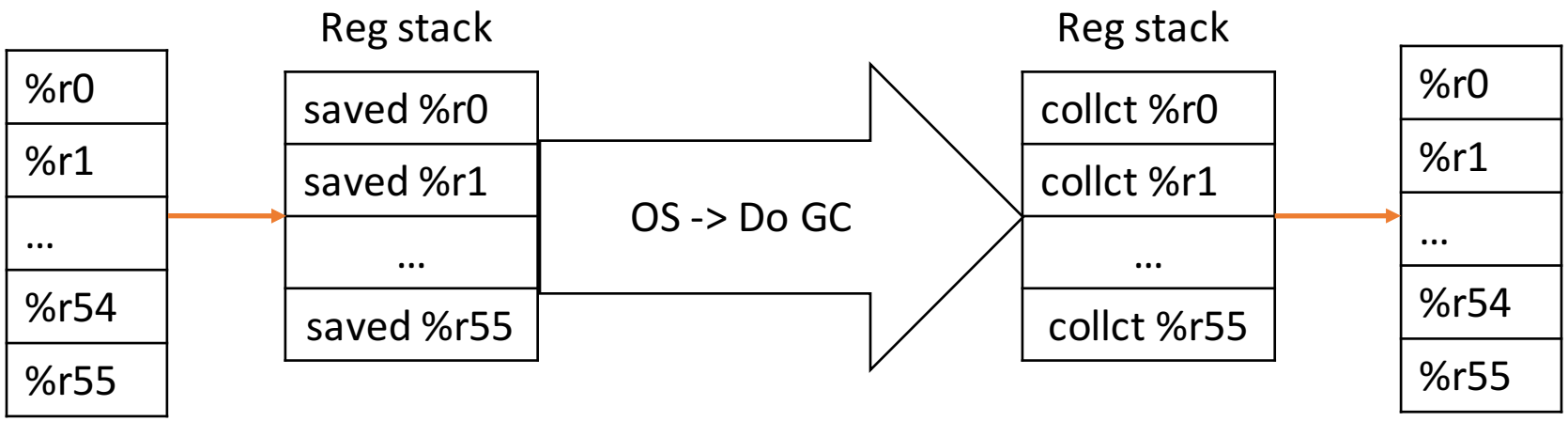
JPoint Student day, 24 апреля 2016

Версия Java

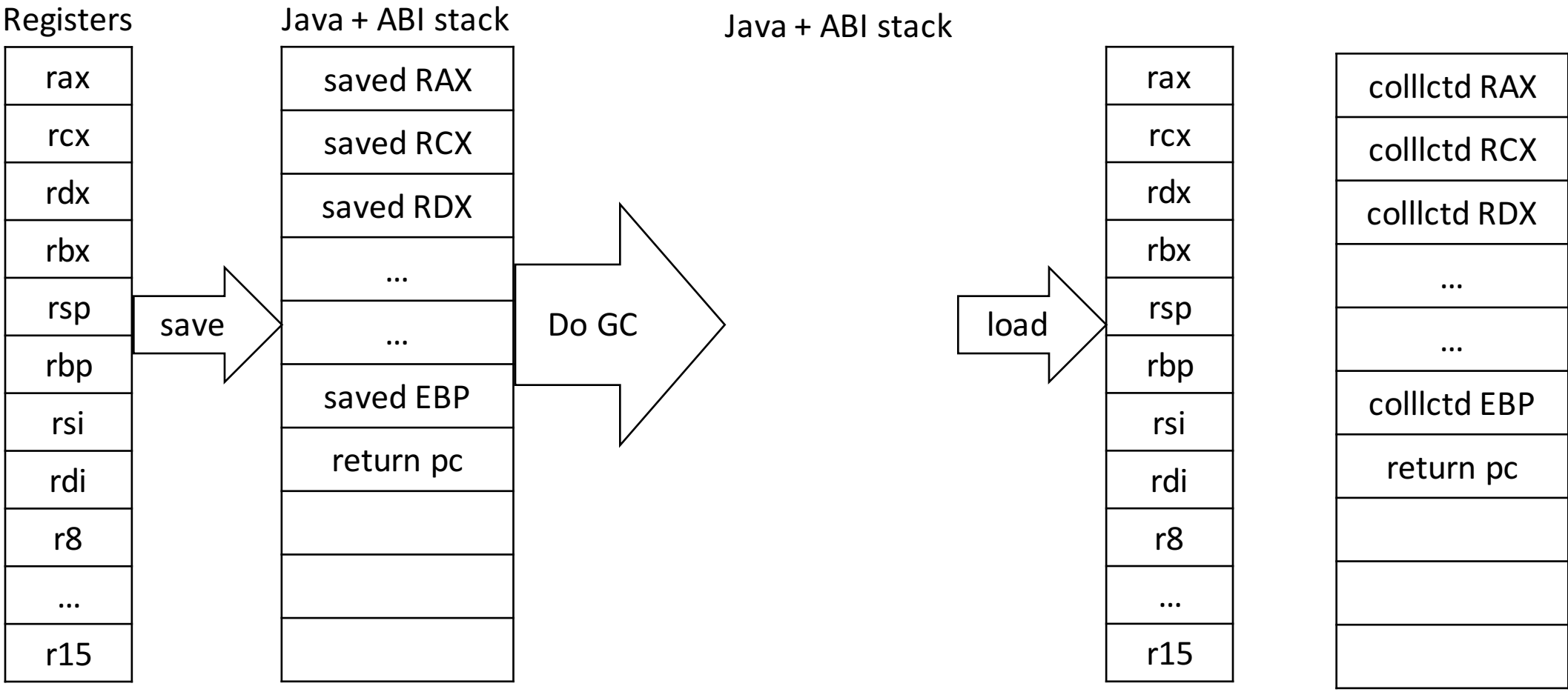
Деоптимизация



GC: Эльбрус


%r0
%r1
…
%r54
%r55
Reg stack
saved %r0
saved %r1
…
saved %r55
OS -> Do GC
Reg stack
collct %r0
collct %r1
…
collct %r55
%r0
%r1
…
%r54
%r55

GC: x86-64


Registers
rax
rcx
rdx
rbx
rsp
rbp
rsi
rdi
r8
…
r15
save
Java + ABI stack
saved RAX
saved RCX
saved RDX
…
…
saved EBP
return pc
Do GC
Java + ABI stack
load
rax
rcx
rdx
rbx
rsp
rbp
rsi
rdi
r8
…
r15
colllctd RAX
colllctd RCX
colllctd RDX
…
…
colllctd EBP
return pc

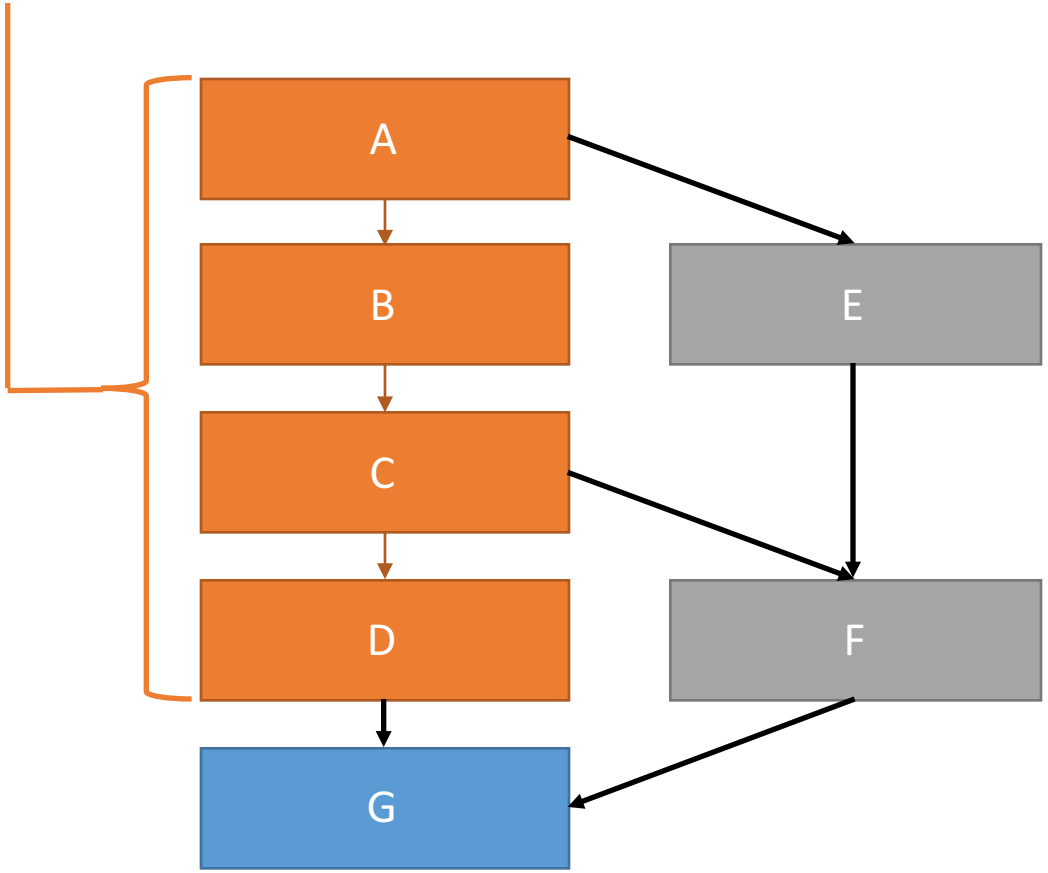
Планирование инструкций

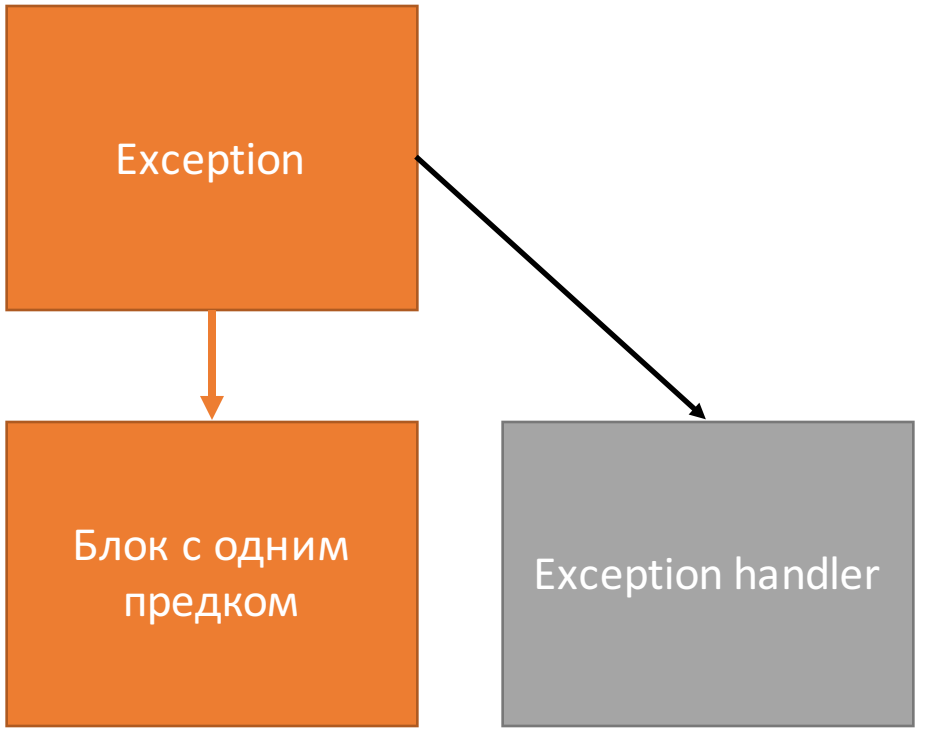
- До 6 арифметических инструкций в широкой команде
- Java имеет короткие базовые блоки
- Списочное планирование дает 1.3 инструкции на широкую команду

+30% производительности

Межблоковое планирование инструкций

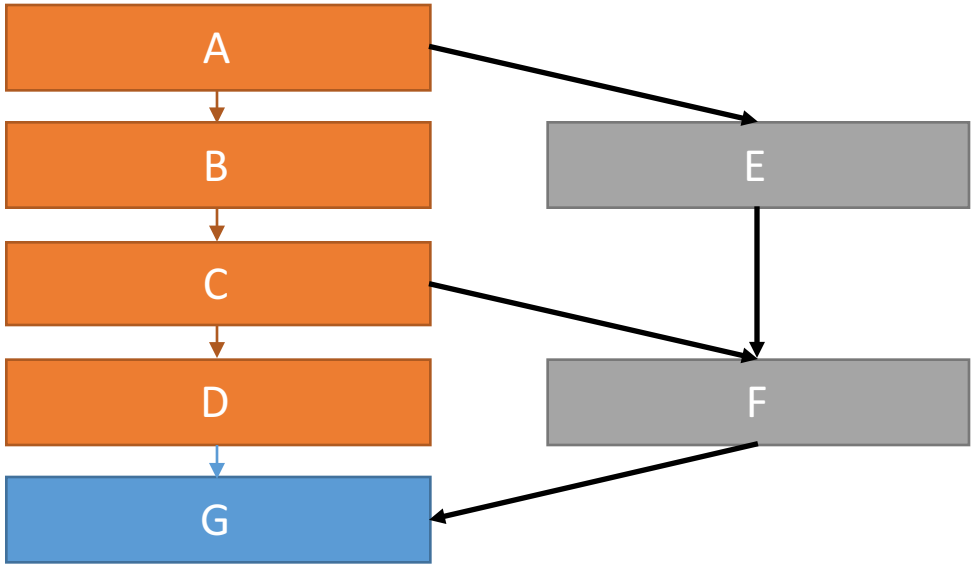
Суперблок

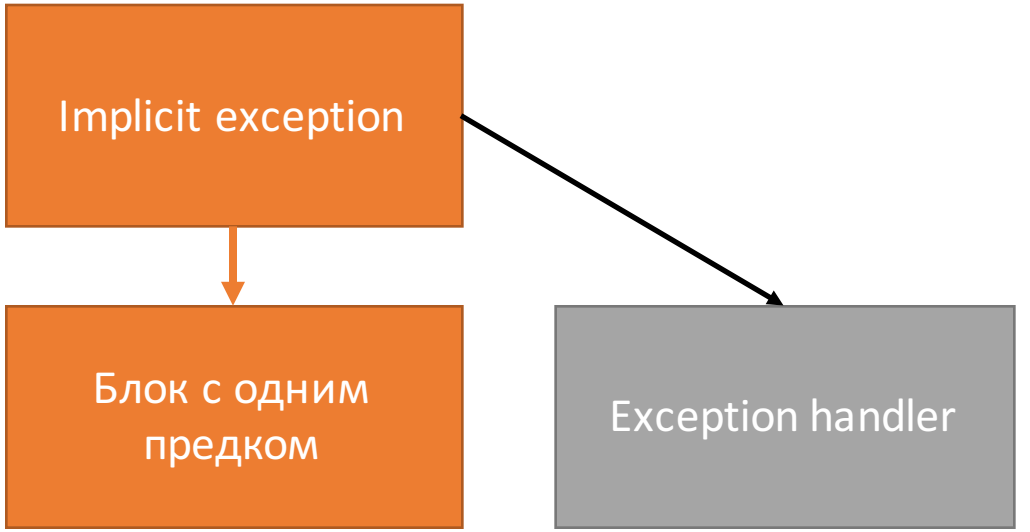




Межблоковое планирование инструкций

- Суперблоки дают еще 15% производительности
- Среднее количество инструкций в широкой команде 2.2, по-прежнему значительно меньше 6

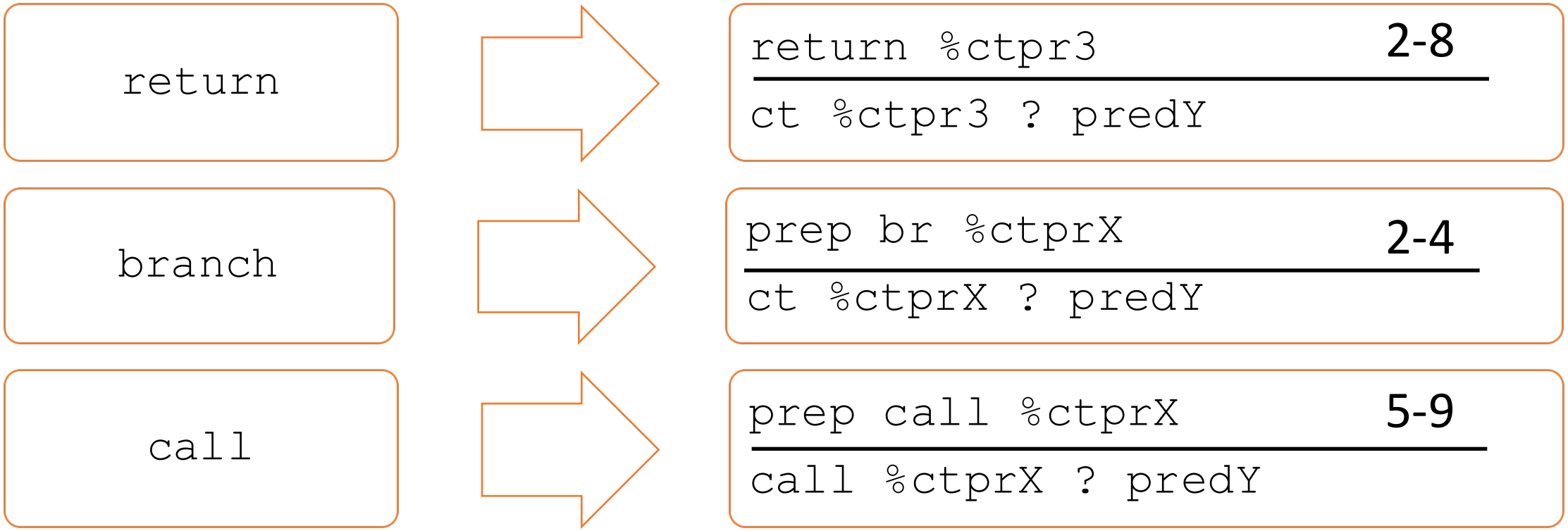




Распределение регистров

- Регистры значения: 192 через регистровое окно, 32 глобальных
- Предикаты: 32, можно создать до 65 потоков управления
- Управляющие регистры: 3, служат для передачи управления

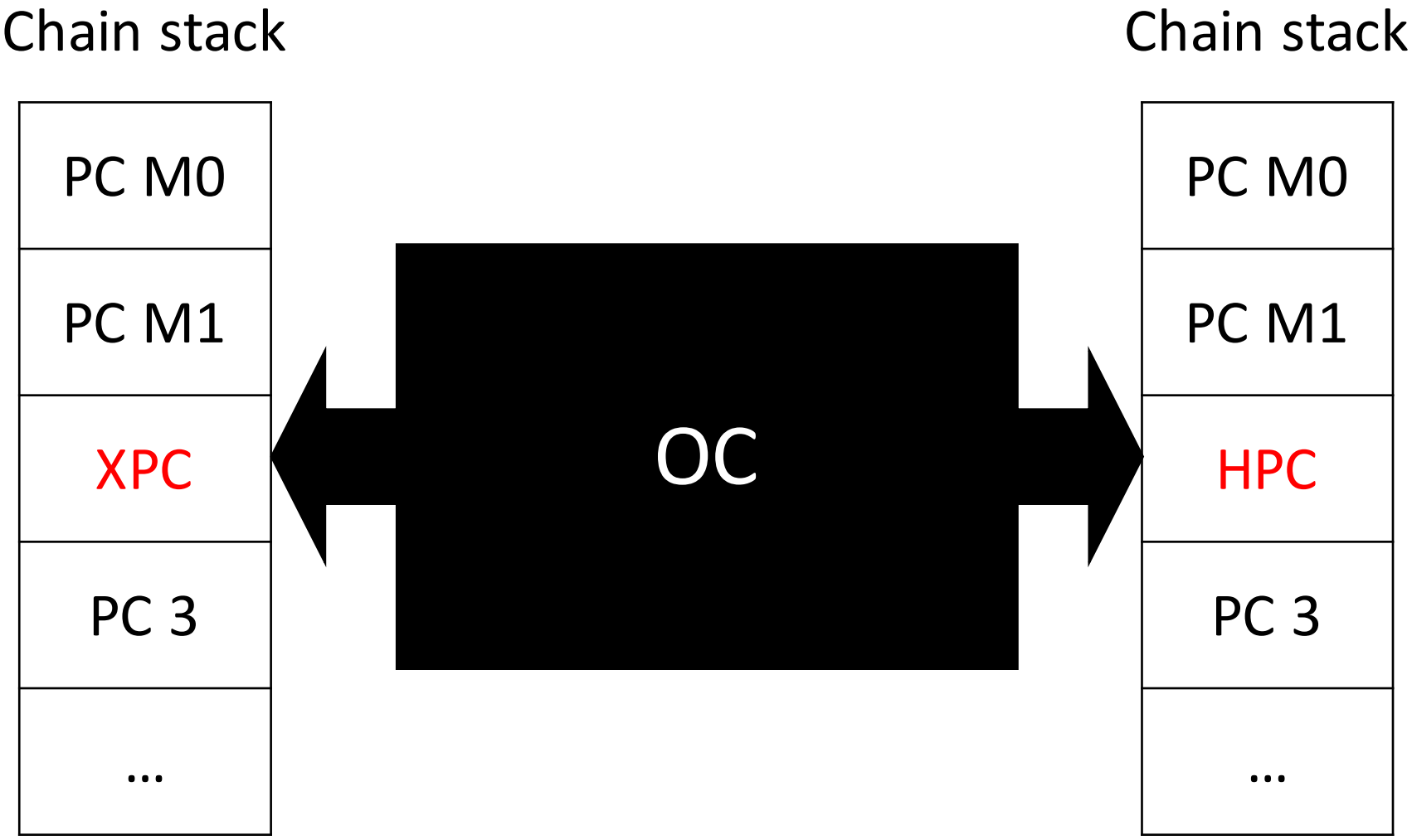
Control Transition Preparation Register



Распределение регистров

- Linear scan register allocator для управляющих регистров во время планирования, +10% производительности
- В будущем планируем объединить распределение регистров и планирование инструкций для всех типов

Проброс исключений



Исключения – Медленно

Исключения – Медленно (пока…)