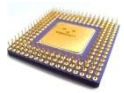
Introduction to MIPS

Andrei Tatarnikov
atatarnikov@hse.ru

National Research University Higher School of Economics (HSE)
http://www.hse.ru

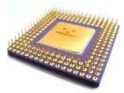
NATIONAL RESEARCH UNIVERSITY



Общие сведения

- Разработан в Стэнфорде группой Джона Хеннесси (проект начат в 1981 году)
- Применяется в основном во встроенных системах
- Следует философии RISC
- Инструкции постоянной длины 32 бита
- Конвейер инструкций из 5 стадий
- Существуют 32 и 64-битные модификации (MIPS32 и MIPS64)

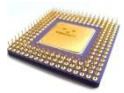
*Регистры MIPS**



- 32 регистра общего назначения (GPR)
- Счетчик команд (PC)
- Регистры умножения и деления (HI и LO)
- 32 регистра плавающей арифметики (FPR)
- 5 управляющих регистров плавающей арифметики (FCR{0, 25, 26, 28}, FCSR)

- ***- Размер всех регистров 32 бита***

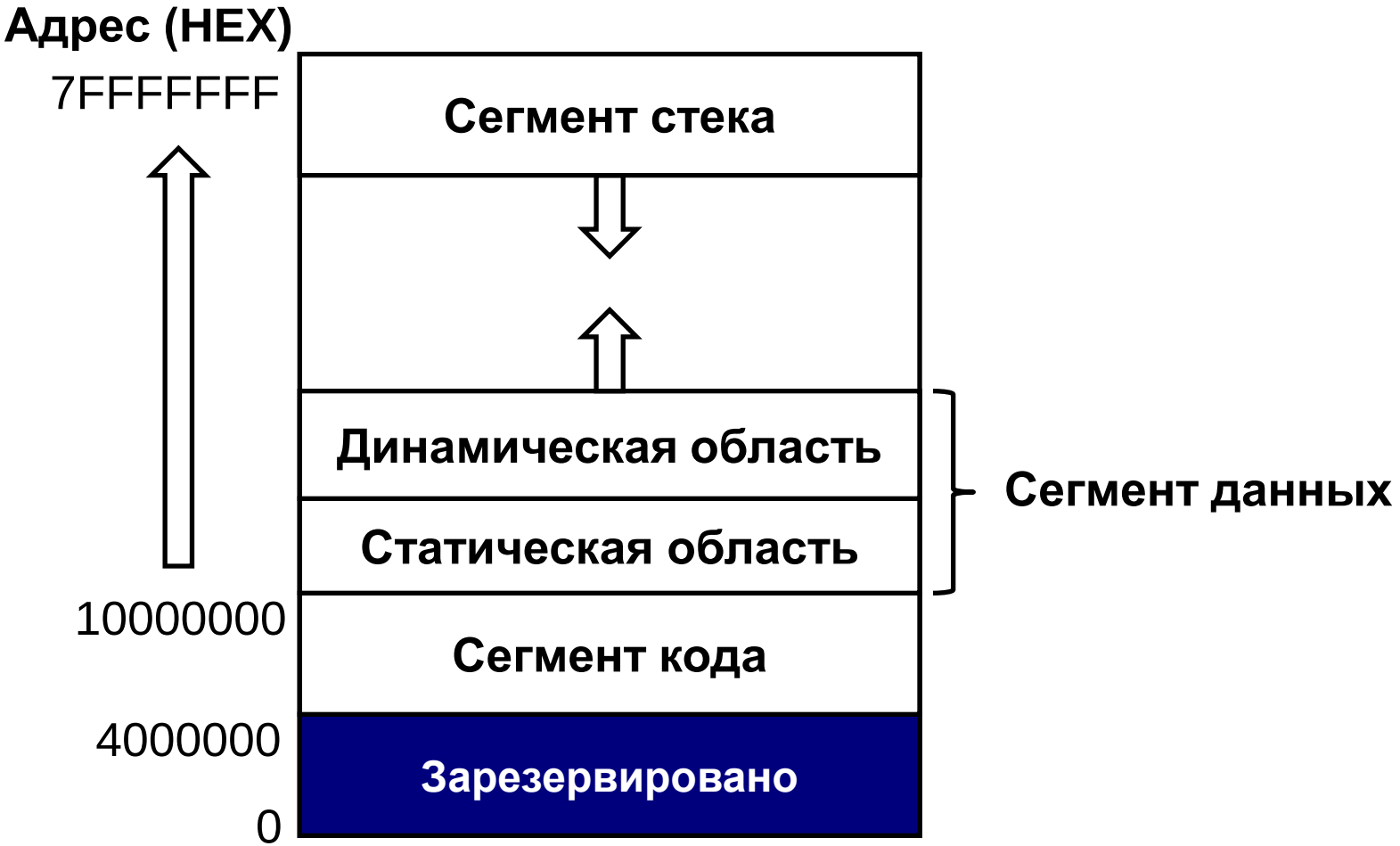
Регистры общего назначения



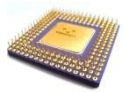
Имя	Номер	Назначение
$zero	0	Всегда хранит 0
$at	1	Зарезервирован для ассемблера
$v1, $v2	2,3	Возвращаемые значения функции
$a0-$a3	4-7	Аргументы функции (с 1-го по 4-й)
$t0-$t7	8-15	Временные, не сохраняются
$s0-$s7	16-23	Сохраняются после вызова функции
$t8, $t9	24, 25	Временные, не сохраняются
$k0, $k1	26, 27	Зарезервированы для ядра ОС
$gp	28	Указатель на область глобальных данных
$sp	29	Указатель стека
$fp	30	Указатель фрейма (если используется) или $s8
$ra	31	Адрес возврата

Структура памяти

Используемые сегменты: code, data, stack



Инструкции MIPS



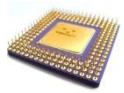
Типы инструкций

- Арифметические (Arithmetic)
- Логические (Logical)
- Передачи данных (Data transfer)
- Ветвления (Condition branch)
- Безусловного перехода (Unconditional jump)

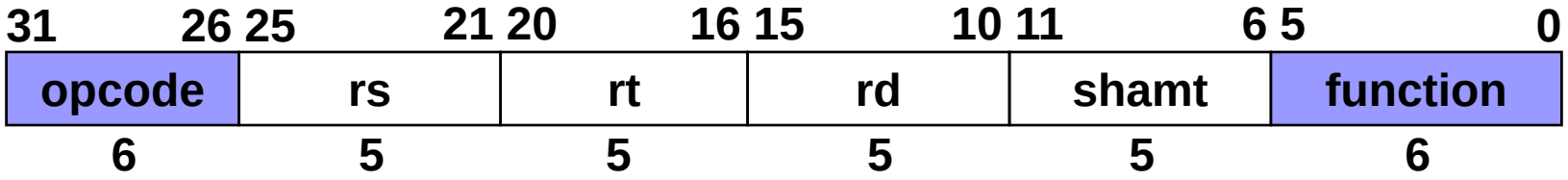
Пример использования: A[12] = x + A[8]

```
lw $t0, 32($s3)    # $t0 := A[8] ($s3 хранит A)
add $t0, $s2, $t0  # $t0 := x + A[8] ($s2 хранит x)
sw $t0, 48($s3)    # A[12] := x + A[8]
```

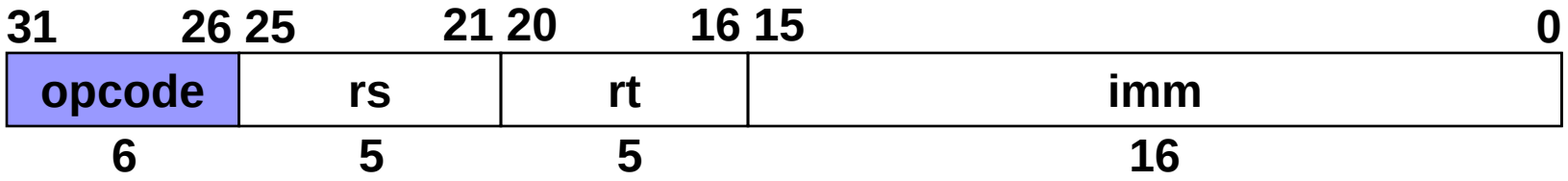
Формат инструкций MIPS



Register (R-Type)



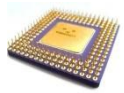
Immediate (I-Type)



Jump (J-Type)

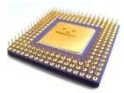
opcode	Код операции	***shamt***	Сдвиг
rs	Регистр-получатель	***function***	Код функции
rt	Регистр-источник	***imm***	Константа
rd	Регистр-источник	***idx***	Индекс инструкции

Основные инструкции (часть 1)



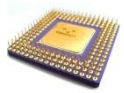
Имя	Ф-т	Тип	Пример	Значение
add	R	Arithmetic	add $s1, $s2, $s3	$s1 = $s2 + $s3
sub	R	Arithmetic	sub $s1, $s2, $s3	$s1 = $s2 - $s3
addi	I	Arithmetic	addi $s1, $s2, 20	$s1 = $s2 + 20
lw	I	Data transfer	lw $s1, 20($s2)	$s1 = Memory[$s2 + 20]
sw	I	Data transfer	sw $s1, 20($s2)	Memory[$s2 + 20] = $s1
and	R	Logical	and $s1, $s2, $s3	$s1 = $s2 & $s3
or	R	Logical	or $s1, $s2, $s3	$s1 = $s2 | $s3
nor	R	Logical	nor $s1, $s2, $s3	$s1 = ~($s2 | $s3)
andi	I	Logical	andi $s1, $s2, 20	$s1 = $s2 & 20
ori	I	Logical	ori $s1, $s2, 20	$s1 = $s2 | 20
sll	R	Logical	sll $s1, $s2, 10	$s1 = $s2 << 10
srl	R	Logical	srl $s1, $s2, 10	$s1 = $s2 >> 10

Основные инструкции (часть 2)



Имя	Ф-т	Тип	Пример	Значение
ll	I	Data transfer	ll $s1, 20($s2)	$s1 = Memory*[$s2 + 20] 1st part of atomic swap
sc	I	Data transfer	sc $s1, 20($s2)	Memory [$s2 + 20] = $s1, $s1 = 0 or 1 2nd part of atomic swap
lui	I	Data transfer	lui $s1, 20	$s1 = 20 * 2^{16}
beq	I	Cond. branch	beq $s1, $s2, 25	if ($s1 == $s2) go to PC+4+100
bne	I	Cond. branch	bne $s1, $s2, 25	if ($s1 != $s2) go to PC+4+100
slt	R	Cond. branch	slt $s1, $s2, $s3	if ($s2 < $s3) $s1=1; else $s1=0
slti	I	Cond. branch	slti $s1, $s2, 20	if ($s2 < 20) $s1=1; else $s1=0
j	J	Uncond. jump	j 2500	go to 10000
jr	R	Uncond. jump	jr $ra	go to $ra
jal	J	Uncond. jump	jr 2500	$ra = PC + 4; go to 10000

Разработка программ для MIPS



Симуляторы:

- **SPIM,** http://spimsimulator.sourceforge.net/
- **MARS,** http://courses.missouristate.edu/kenvollmar/mars/

Формат программы:

```
        .data
item:
        .word 1          # Глобальные данные
        .text
        .globl main      # Объявляется глобальной
main:
        lw $t0, item     # Пользовательский код
        …
```

Компиляция программ на Си в код для MIPS

Язык Си:

```
void swap(int v[], int k)
{
  int temp;
  temp = v[k];
  v[k] = v[k+1];
  v[k+1] = temp;
}
```

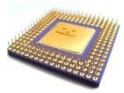
Ассемблер MIPS:

```
swap:
  sll $t1, $a1, 2
  add $t1, $a0, $t1
  lw $t0, 0($t1)
  lw $t2, 4($t1)
  sw $t2, 0($t1)
  sw $t0, 4($t1)
  jr $ra
```

Машинный код для MIPS:

```
00000000101000010000000000011000
00000000000011000000110000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```

Ветвления в MIPS

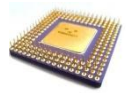


Язык Си:

```
if (i == j)
    f = g + h;
else
    f = g - h;
```

MIPS (i -> $s3, j -> $s4, f -> $s0, g -> $s1, h -> $s2):

```
    bne $s3,$s4, else       # go to Else if i not = j
    add $s0,$s1,$s2         # f = g + h
    j exit                  # jump out of the if
else:
    sub $s0,$s1,$s2         # f = g - h
exit:
```

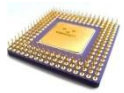


Работа со стеком в MIPS

```
addi $sp,$sp,-12
sw $t1, 8($sp)
sw $t0, 4($sp)
sw $s0, 0($sp)

add $t0,$a0,$a1
add $t1,$a2,$a3
sub $s0,$t0,$t1
```

```
add $v0,$s0,$zero

lw $s0, 0($sp)
lw $t0, 4($sp)
lw $t1, 8($sp)
addi $sp,$sp,12

jr $ra
```

Простая программ для MIPS



```
    .data
success_msg:
    .asciiz "Success!"
failed_msg:
    .asciiz "Failed!"
    .text
    .globl main
main:
    addiu $s0, $zero, 2
    addiu $s1, $zero, 3
    addiu $s2, $zero, 5
    add   $t0, $s0, $s1
    bne   $t0, $s2, fail
    sub $t1, $s2, $s1
    bne $t1, $s0, fail
    la $a0, success_msg
    li $v0,4
    syscall
exit:
    li $v0,10
    syscall
fail:
    la $a0, failed_msg
    li $v0,4
    syscall
    j exit
```