

# S3 (6) - Cod Cool - MIPS

Tuesday, January 10, 2023 1:30 PM

## 1. DEFAULT DISPATCH TABLES

```

Object_dispTab:
    .word Object.abort
    .word Object.type_name
    .word Object.copy
IO_dispTab:
    .word Object.abort
    .word Object.type_name
    .word Object.copy
    .word IO.out_string
    .word IO.out_int
    .word IO.in_string
    .word IO.in_int
Int_dispTab:
    .word Object.abort
    .word Object.type_name
    .word Object.copy
String_dispTab:
    .word Object.abort
    .word Object.type_name
    .word Object.copy
    .word String.length
    .word String.concat
    .word String.substr
Bool_dispTab:
    .word Object.abort
    .word Object.type_name
    .word Object.copy
Main_dispTab:
    .word Object.abort
    .word Object.type_name
    .word Object.copy
    .word Main.main
    .globl heap_start
    
```

## 2. ORGANIZARE OBIECT $\Rightarrow$ \$00 self objct

Reprezentarea obiectelor în memorie I

	Offset	
...		
Etichetă clasă	0	$\$00 + 0 \Rightarrow 0 (\$00)$
Dimensiune obiect	4	$\$00 + 4 \Rightarrow 4 (\$04)$
Dispatch pointer	8	$\$00 + 8 \Rightarrow 8 (\$08)$
Atribut 1	12	$\$00 + 12 \Rightarrow 12 (\$0C)$
Atribut 2	16	$\$00 + 16 \Rightarrow 16 (\$10)$
...		...

## 3. ÎNREGISTRAREA DE ACTIVARE

Înregistrarea de activare

Conținut	Adresă
Parametru $n$	
$\vdots$	$\vdots$
Parametru 2	$\$fp + 16$
Parametru 1	$\$fp + 12$
$\$fp$	
$\$s0$	
$\$ra$	$\$fp$
	$\$sp$

Înregistrarea de activare cu variabile LET

Conținut	Adresă
Parametru $n$	
$\vdots$	$\vdots$
Parametru 2	$\$fp + 16$
Parametru 1	$\$fp + 12$
$\$fp$	
$\$s0$	
$\$ra$	$\$fp$
Variabilă let 1	$\$fp - 4$
Variabilă let 2	$\$fp - 8$
$\vdots$	$\vdots$
Variabilă let $m$	
	$\$sp$

$\Rightarrow$  param 1  $\Rightarrow$   $\$fp + 12$  12( $\$fp$ )  
 param 2  $\Rightarrow$   $\$fp + 16$  16( $\$fp$ )  
 ...

let Var 1  $\Rightarrow$   $\$fp - 4$  -4( $\$fp$ )  
 let Var 2  $\Rightarrow$   $\$fp - 8$  -8( $\$fp$ )  
 ...

## EXERCITIU

①

```
1 class A {
2     content : String <- "abc";
3     f(str : String) : String {
4         content <- str.concat(content);
5     };
6 }
```

```
lw $a0 12($s0) } push content
sw $a0 0($sp)
addiu $sp $sp -4
lw $a0 12($fp) => get dispatch object (str)
<verificare dispatch on void>
lw $t1 8($a0) => dispatch
lw $t1 16($t1) => method offset (concat)
jalr $t1
```

str = primul param al lui f => \$fp+1

concat = offset 16 in thing - dispatch

content = primul atribut al clasei => \$s0+12

! OBS -> la apelul de functie

① ne pun param. in ordine inversa pe stiva =>

```
gen(param m)
sw $a0 0($sp)
addiu $sp $sp -4 } push
```

② ne inlocuim directul pe care ne face dispatch (ex: e.f())

• self => move \$a0 \$s0

• direct static (ex: "abc") => la \$a0 str\_const1

• Id -> atribut de clasa => <offset> (\$s0)  
param de functie => <offset> (\$fp)  
variabila locala => -<offset> (\$fp)

```
str_const1:
.word 2
.word 5
.word String_dispatch
.word int_const2
.asciiz "abc"
.align 2
```

...  
• rezultatul evaluării unei expresii

③ verificare dispatch on void

④ în care adresă dispatch\_Tabelle în \$t1

① statică: e(A.f()) => la \$t1 A\_dispatch

② dinamică: e.f() => lw \$t1 8(\$a0) => dispatch este mereu la offset 8 fata de \$s0/\$a0

⑤ în care offset metoda (din dispatch)

lw \$t1 <offset> (\$t0)

! Nu uitati ca + dispatch contine metodele lui Object + alte clase moștenite si alina apoi metodele proprii

2

```
class A inherits IO {};

class B inherits A {
  f() : A {{
    self@A.out_string("def");
  }};
};
```

```
A_dispTab:
.word 0 Object.abort
.word 4 Object.type_name
.word 8 Object.copy
.word 12 IO.out_string
.word 16 IO.out_int
.word 20 IO.in_string
.word 24 IO.in_int
```

```
la $a0 str_const6 => "def"
sw $a0 0($sp) } push "def"
addiu $sp $sp -4
move $a0 $s0 => dispatch obj = self
<verificare dispatch on void>
la $t1 A_dispTab => static dispatch
lw $t1 12($t1) => method offset out_string in A_dispTab
jalr $t1
out_string
```

3

```
class A {
  v1 : Int;
  v2 : C;
  f(i : Int, c : C) : SELF_TYPE {
    {
      v1 <- i;
      v2 <- c;
      self;
    }
  };
};

class B inherits A {
  index : Int;
  set(i : Int) : SELF_TYPE {{
    index <- i;
    self;
  }};
};

class C { };

class Main inherits IO {
  main() : Object {
    {
      let b : B <- new B.f(0, new C)
      in b.set(0);
      new B.set(0);
    }
  };
};
```

Se propune următoarea seq. MIPS pt. apelul B.f(0, new C)  
Este corectă? Dacă nu, atunci care este seq. corectă care îi corespunde?

```
la $a0 int_const0 }
sw $a0 0($sp) } push 0
addiu $sp $sp -4
la $a0 B_protObj } new B
jal Object.copy
jal B_init
<verificare dispatch on void>
lw $t1 8($a0) # dispatch table
lw $t1 16($t1) # method offset
jalr $t1 } ret
```

```
B_dispTab:
.word 0 Object.abort
.word 4 Object.type_name
.word 8 Object.copy
.word 12 A.f
.word 16 B.set
```

=> new B.set(0)