

CPSC 213 Lab 2

CPU and Static Variables

Randy Zhu

Upcoming Deadlines

- Assignment 2 due this Friday, January 23rd
- Quiz 1 runs next week January 26-30th in CBTF
 - Reservations already open on <https://us.prairietest.com/>

Helpful Resources

Course Resources

Here, in one place, are all of the key resources for the course.

- [Syllabus](#)
- [Schedule](#)
- [Marks by Learning Goal](#)
- [SM213 Instruction Set Architecture](#)
- [Lecture Notes Companion](#)
- [Setup Companion](#)
- [Assignment Companion](#)
- [Simulator Executable \(Reference Simulator\)](#)
- Simulator Source Code
 - [IntelliJ](#)
 - [Visual Studio Code](#)
- [Hitchhikers Guide to Labs and Assignments](#)
- [How to Prepare for \(and Survive\) your Exams](#)

Save & Grade **Save only**

Operation	Machine Language	Semantics / RTL	Assembly
load immediate	0d--vvvvvvvv	$r[d] \leftarrow v$	ld \$v,rd
load base+offset	1psd	$r[d] \leftarrow m[(o = p \times 4) + r[s]]$	ld o(rs),rd
load indexed	2sid	$r[d] \leftarrow m[r[s] + r[i] \times 4]$	ld (rs,ri,4),rd
store base+offset	3spd	$m[(o = p \times 4) + r[d]] \leftarrow r[s]$	st rs,o(rd)
store indexed	4sdi	$m[r[d] + r[i] \times 4] \leftarrow r[s]$	st rs,(rd,ri,4)
halt	F0--	(stop execution)	halt
nop	FF--	(do nothing)	nop
rr move	60sd	$r[d] \leftarrow r[s]$	mov rs,rd
add	61sd	$r[d] \leftarrow r[d] + r[s]$	add rs,rd
and	62sd	$r[d] \leftarrow r[d] \& r[s]$	and rs,rd
inc	63-d	$r[d] \leftarrow r[d] + 1$	inc rd
inc addr	64-d	$r[d] \leftarrow r[d] + 4$	inca rd
dec	65-d	$r[d] \leftarrow r[d] - 1$	dec rd
dec addr	66-d	$r[d] \leftarrow r[d] - 4$	deca rd
not	67-d	$r[d] \leftarrow \sim r[d]$	not rd
shift	7dss ($ss > 0$) 7dss ($ss < 0$)	$r[d] \leftarrow r[d] << (v = ss)$ $r[d] \leftarrow r[d] >> (v = -ss)$	shl \$v,rd shr \$v,rd
branch	8-pp	$pc \leftarrow (a = pc + p \times 2)$	br a
branch if equal	9rpp	if $r[r] == 0$: $pc \leftarrow (a = pc + p \times 2)$	beq rr,a
branch if greater	Arpp	if $r[r] > 0$: $pc \leftarrow (a = pc + p \times 2)$	bgt rr,a
jump	B---aaaaaaaa	$pc \leftarrow a$	j a
get program counter	6Fpd	$r[d] \leftarrow pc + (o = 2 \times p)$	gpc \$o,rd
jump indirect	Cdpp	$pc \leftarrow r[d] + (o = 2 \times p)$	j o(rd)
jump double ind, b+off	Ddpp	$pc \leftarrow m[(o = 4 \times p) + r[d]]$	j *o(rd)
jump double ind, index	Edi-	$pc \leftarrow m[4 \times r[i] + r[d]]$	j *(rd,ri,4)
system call	F1nn	* See section on next page	sys \$n

Static vs Dynamic

Q1

- Address of a global variable in C
 - Static
- Address of an instance variable in Java
 - Dynamic

Static vs Dynamic

Q1

- Value of a global variable in C
 - Dynamic
- Address of a function in C
 - Static

RTL to SM213 Assembly

.long = 4 bytes

	0x1000	0x0
	0x2000	0x0
	0x2004	0x0
	0x2004	0x0
	0x2004	0x0

Assembly Code:

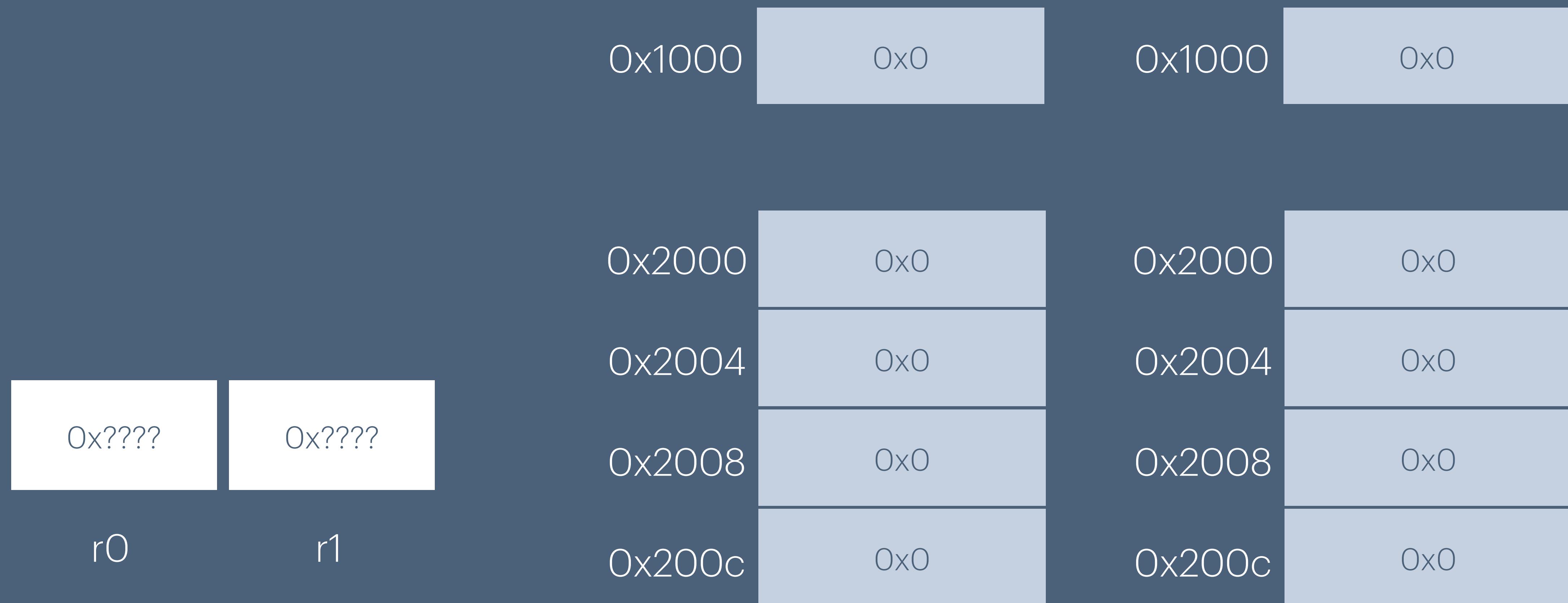
```
.pos 0x1000
a: .long 0

.pos 0x2000
b: .long 0
    .long 0
    .long 0
    .long 0 # enough times
```

Note that each sub-section above

RTL 1

Memory Behaviour



RTL 1

Memory Behaviour

$r0 \leq 0xafe$

0x????

r0

0x????

r1

0x1000

0x0

0x1000

0x0

0x2000

0x0

0x2000

0x0

0x2004

0x0

0x2004

0x0

0x2008

0x0

0x2008

0x0

0x200c

0x0

0x200c

0x0

RTL 1

Memory Behaviour

$r0 \leq 0xafe$

0xafe

r0

0x????

r1

0x1000

0x0

0x1000

0x0

0x2000

0x0

0x2000

0x0

0x2004

0x0

0x2004

0x0

0x2008

0x0

0x2008

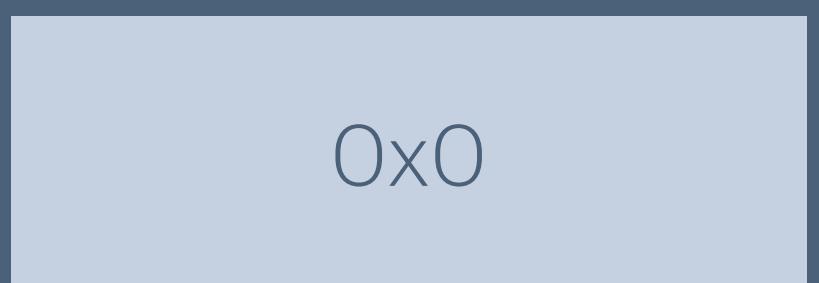
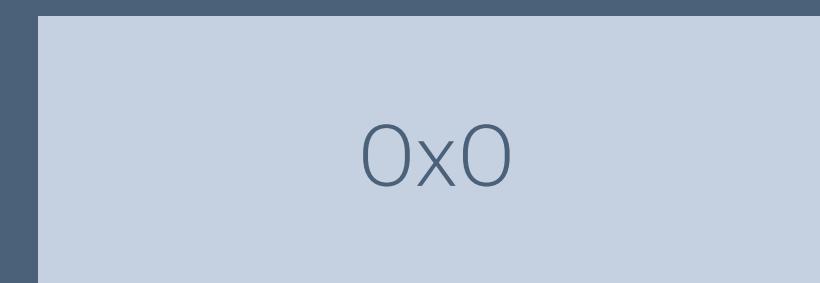
0x0

0x200c

0x0

0x200c

0x0



RTL 1

Memory Behaviour

$r0 \leq 0xafe$

$r1 \leq 0x2000$

0xafe

0x????

$r0$

$r1$

0x1000

0x0

0x1000

0x0

0x2000

0x0

0x2000

0x0

0x2004

0x0

0x2004

0x0

0x2008

0x0

0x2008

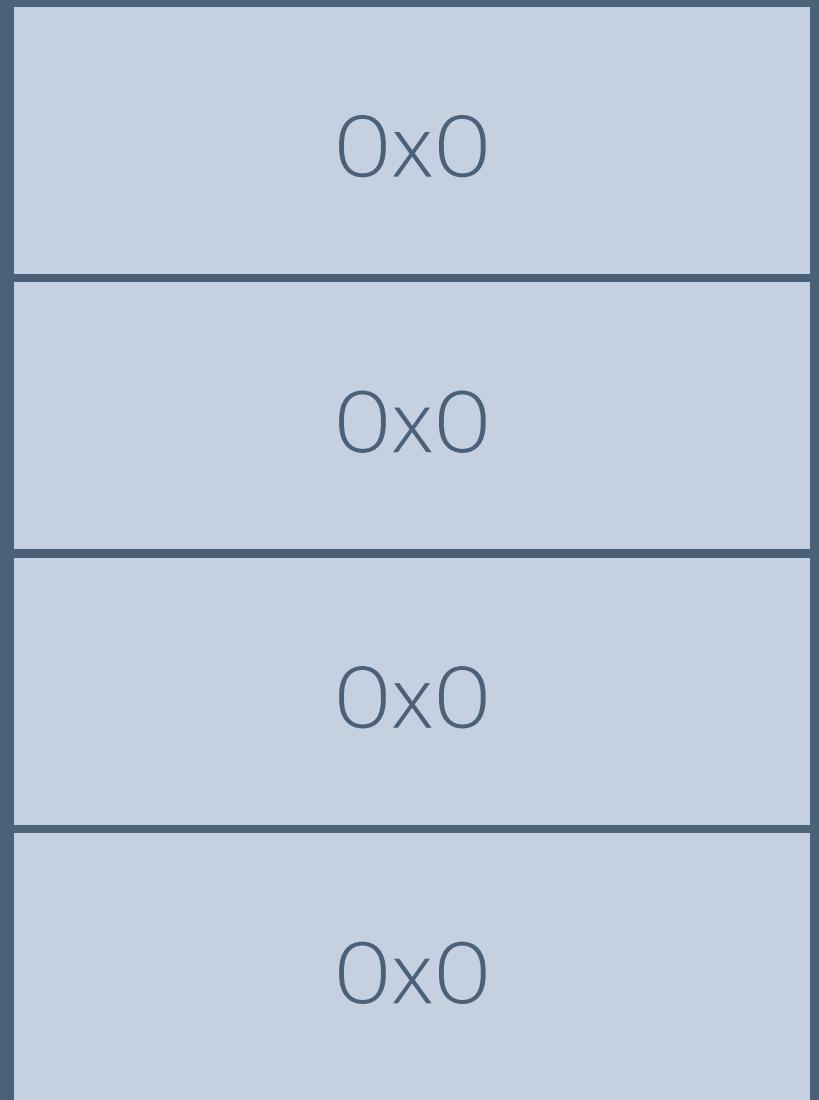
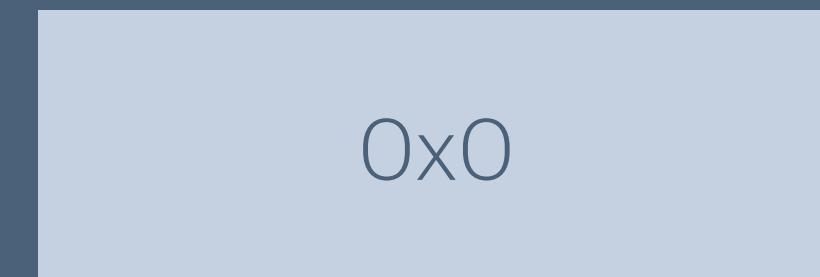
0x0

0x200c

0x0

0x200c

0x0



RTL 1

Memory Behaviour

$r0 \leq 0xafe$

$r1 \leq 0x2000$

0xafe

0x2000

$r0$

$r1$

0x1000

0x0

0x1000

0x0

0x2000

0x0

0x2000

0x0

0x2004

0x0

0x2004

0x0

0x2008

0x0

0x2008

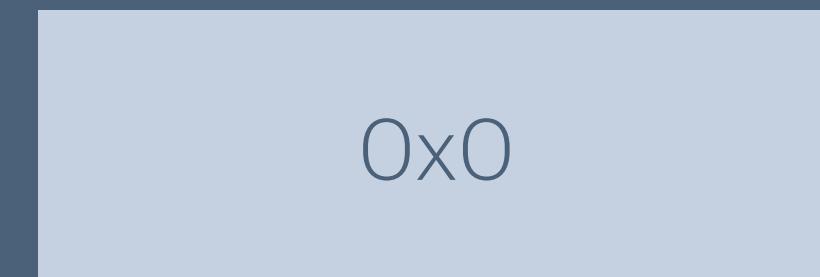
0x0

0x200c

0x0

0x200c

0x0



RTL 1

Memory Behaviour

$r0 \leq 0xafe$

0x1000

0x0

0x1000

0x0

$r1 \leq 0x2000$

$m[0xc + r1] \leq r0$

0x2000

0x0

0x2000

0x0

0xafe

0x2000

0x2004

0x0

0x2004

0x0

0x2008

0x0

0x2008

0x0

0x200c

0x0

0x200c

0x0

$r0$

$r1$

RTL 1

Memory Behaviour

$r0 \leq 0xafe$

0x1000

0x0

0x1000

0x0

$r1 \leq 0x2000$

$m[0xc + r1] \leq r0$

0x2000

0x0

0x2000

0x0

0xafe

0x2000

0x2004

0x0

0x2004

0x0

0x2008

0x0

0x2008

0x0

0x200c

0x0

0x200c

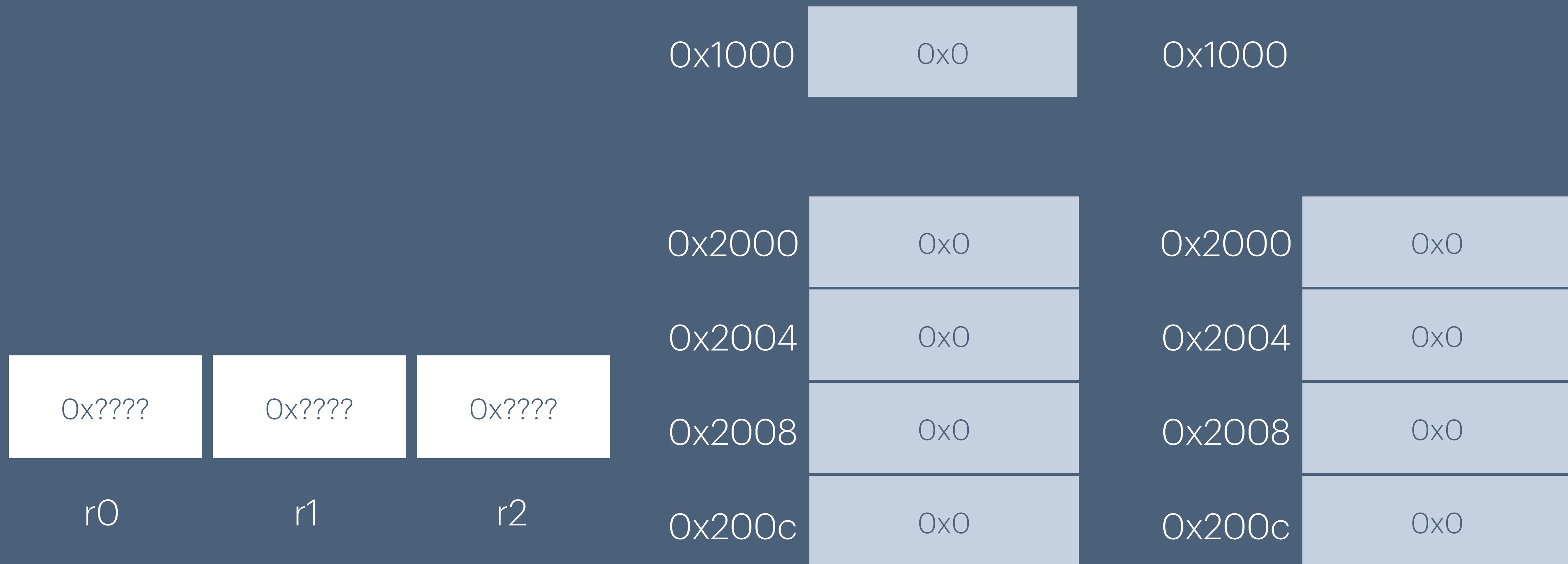
0xafe

$r0$

$r1$

RTL 2

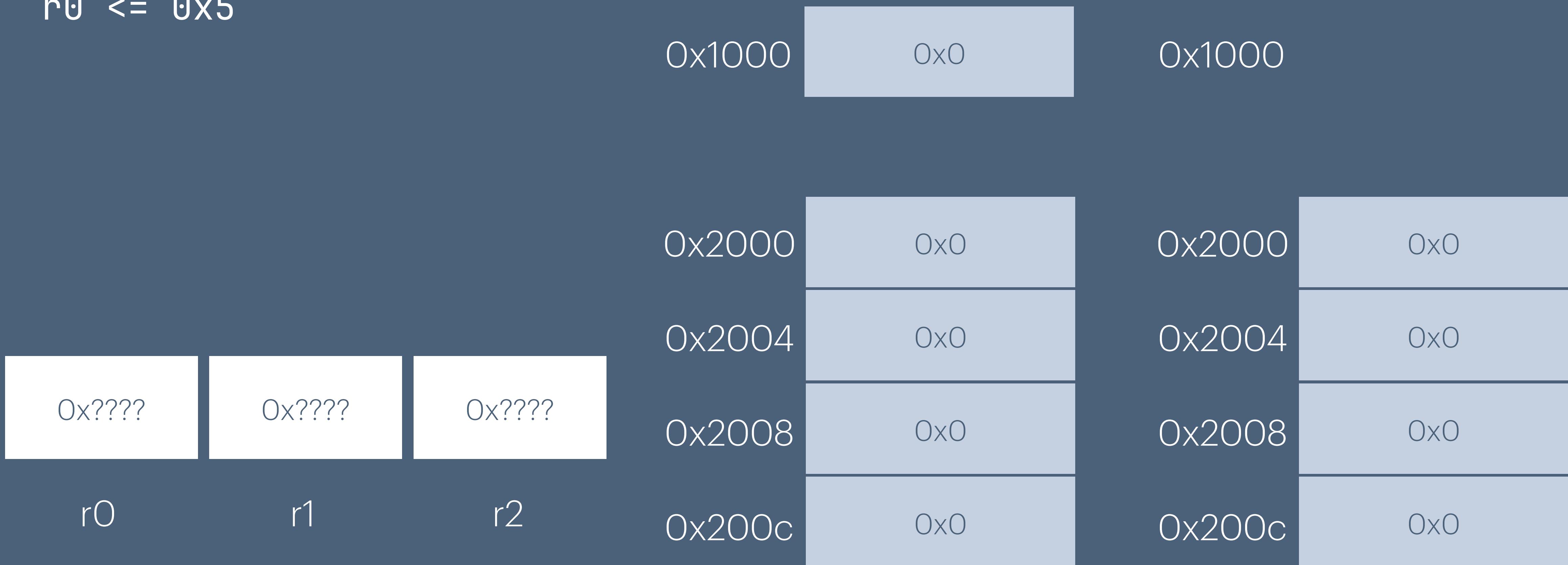
Memory Behaviour



RTL 2

Memory Behaviour

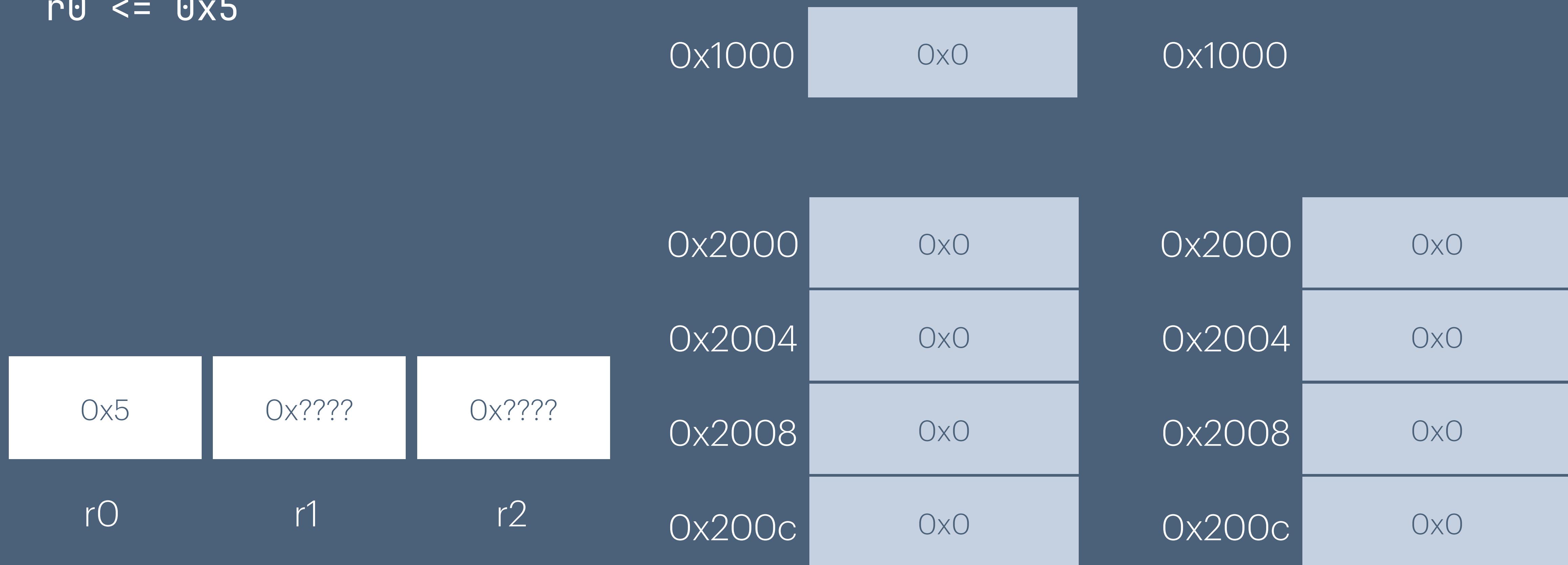
$r0 \leq 0x5$



RTL 2

Memory Behaviour

$r0 \leq 0x5$



RTL 2

Memory Behaviour

$r0 \leq 0x5$

$r1 \leq 0x1000$

0x5

0x????

0x????

$r0$

$r1$

$r2$

0x1000

0x0

0x1000

0x2000

0x0

0x2000

0x0

0x2004

0x0

0x2004

0x0

0x2008

0x0

0x2008

0x0

0x200c

0x0

0x200c

0x0

0x1000

0x2000

0x2004

0x2008

0x200c

0x2000

0x2004

0x2008

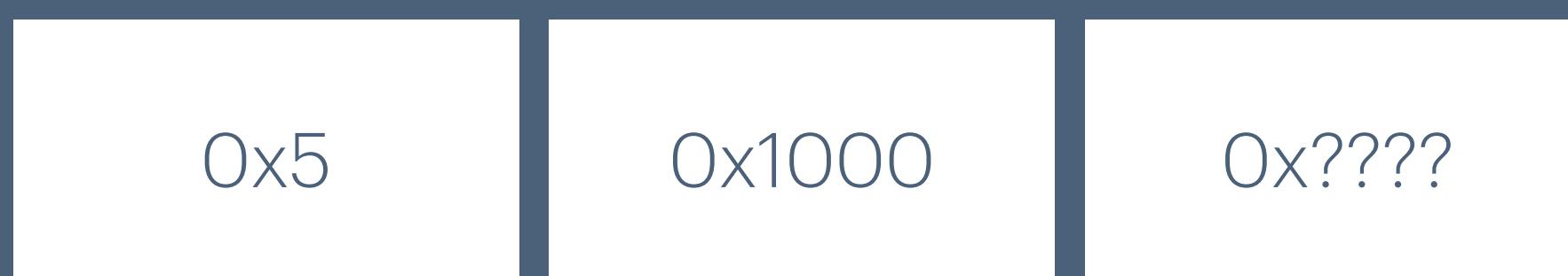
0x200c

RTL 2

Memory Behaviour

$r0 \leq 0x5$

$r1 \leq 0x1000$



0x1000 0x0 0x1000

0x2000 0x0 0x2000 0x0

0x2004 0x0 0x2004 0x0

0x2008 0x0 0x2008 0x0

0x200c 0x0 0x200c 0x0

r0

r1

r2

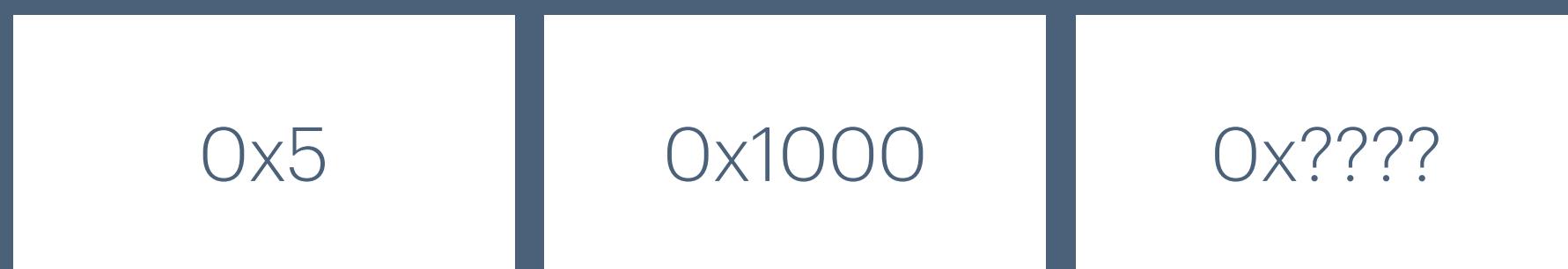
RTL 2

Memory Behaviour

$r0 \leq 0x5$

$r1 \leq 0x1000$

$r2 \leq m[r1]$



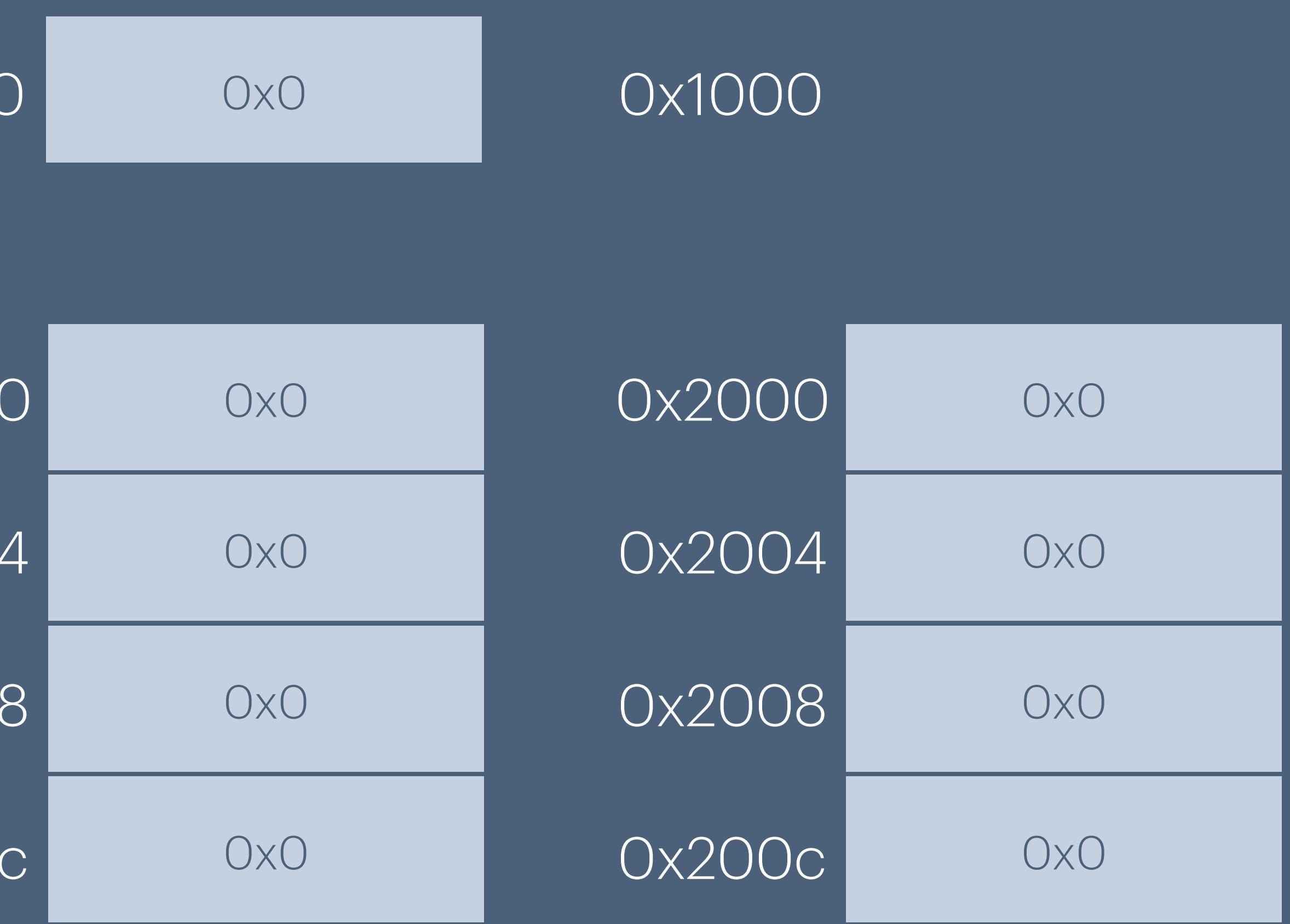
0x1000 0x0 0x1000

0x2000 0x0 0x2000

0x2004 0x0 0x2004

0x2008 0x0 0x2008

0x200c 0x0 0x200c



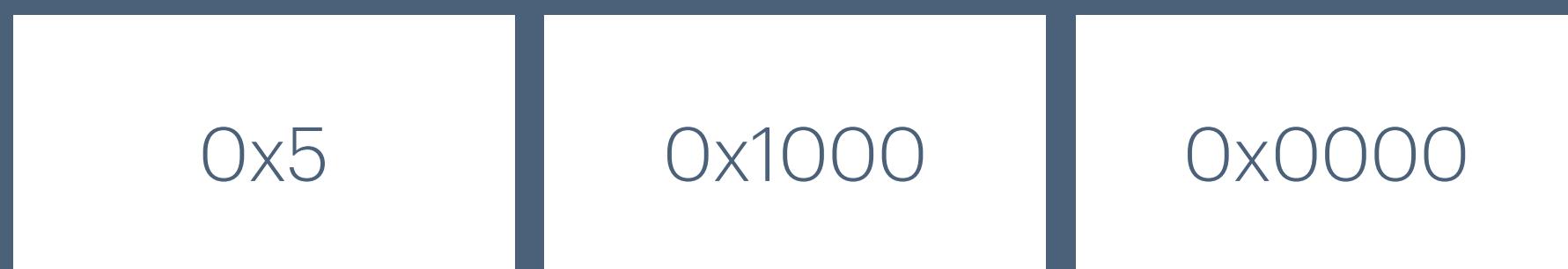
RTL 2

Memory Behaviour

$r0 \leq 0x5$

$r1 \leq 0x1000$

$r2 \leq m[r1]$



$r0$

$r1$

$r2$

0x1000

0x0

0x1000

0x2000

0x0

0x2000

0x2004

0x0

0x2004

0x2008

0x0

0x2008

0x200c

0x0

0x200c

0x0

0x0

0x0

0x0

RTL 2

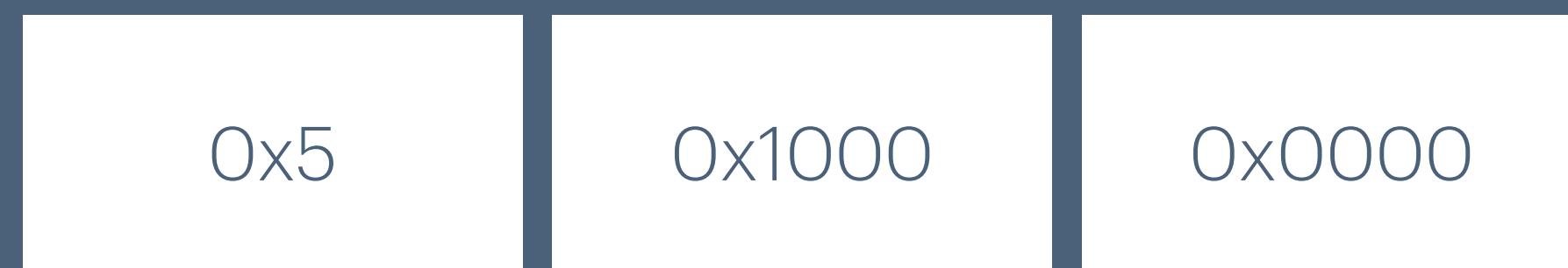
Memory Behaviour

$r0 \leq 0x5$

$r1 \leq 0x1000$

$r2 \leq m[r1]$

$r0 \leq \sim r0$



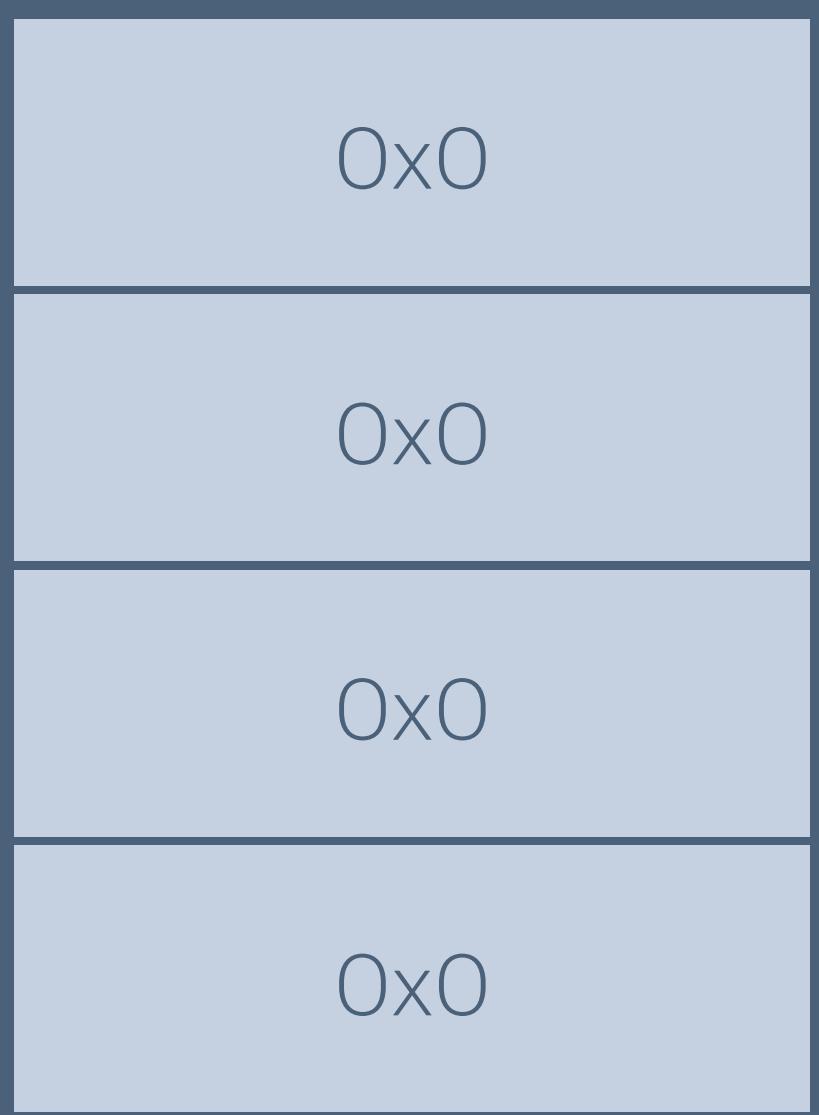
0x1000 0x0 0x1000

0x2000 0x0 0x2000

0x2004 0x0 0x2004

0x2008 0x0 0x2008

0x200c 0x0 0x200c



RTL 2

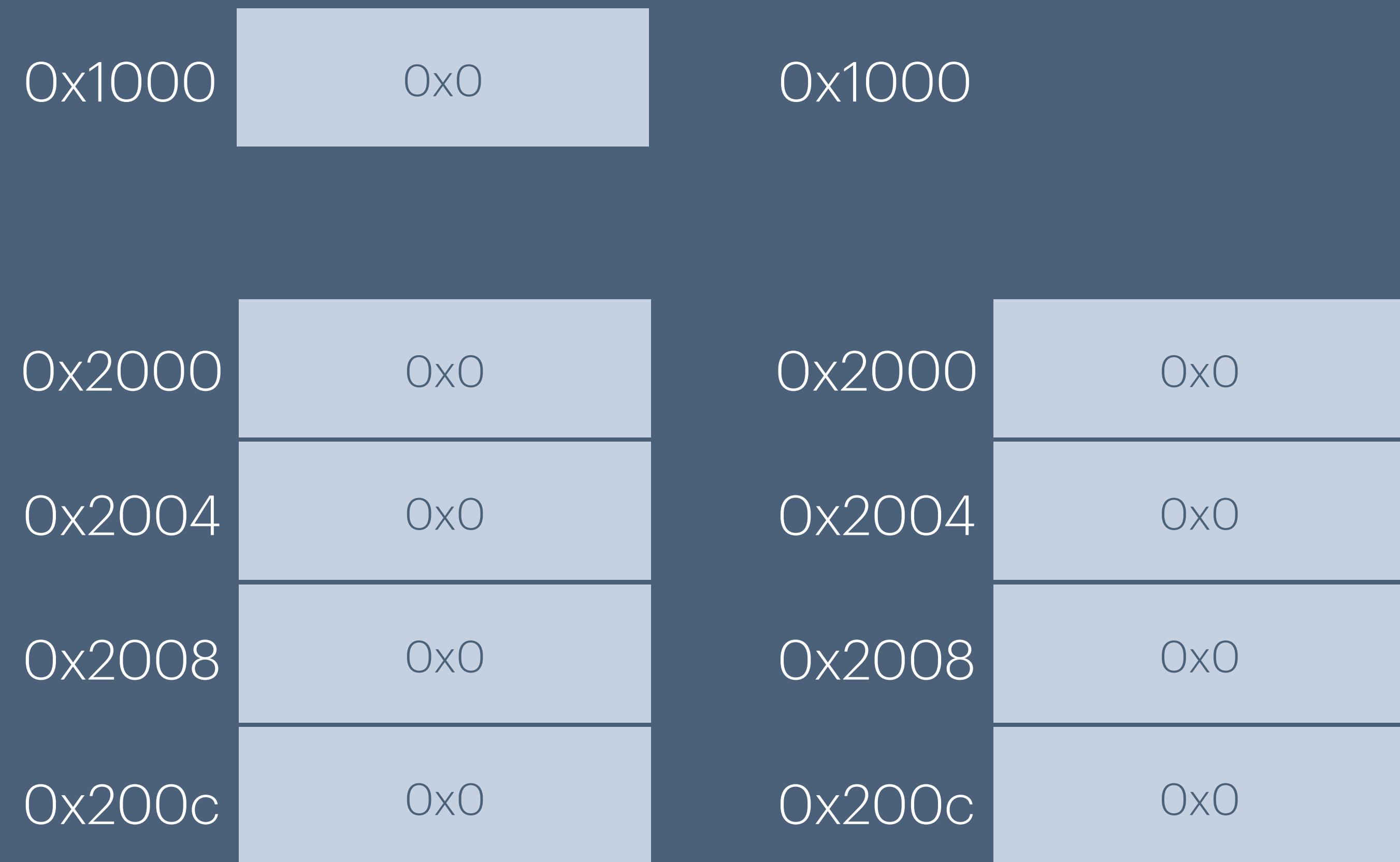
Memory Behaviour

$r0 \leq 0x5$

$r1 \leq 0x1000$

$r2 \leq m[r1]$

$r0 \leq \sim r0$



RTL 2

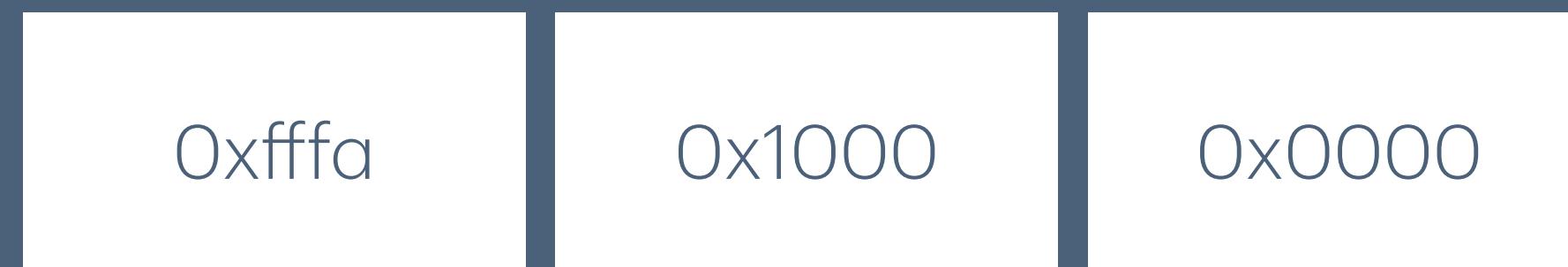
Memory Behaviour

$r0 \leq 0x5$ $r0 \leq r0 + 1$

$r1 \leq 0x1000$

$r2 \leq m[r1]$

$r0 \leq \sim r0$



r0

r1

r2

0x1000

0x0

0x1000

0x2000

0x0

0x2000

0x0

0x2004

0x0

0x2004

0x0

0x2008

0x0

0x2008

0x0

0x200c

0x0

0x200c

0x0

RTL 2

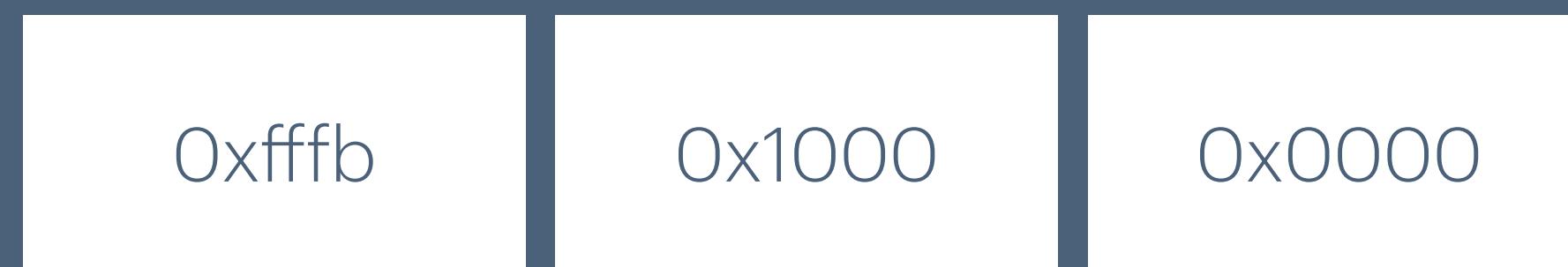
Memory Behaviour

$r0 \leq 0x5$ $r0 \leq r0 + 1$

$r1 \leq 0x1000$

$r2 \leq m[r1]$

$r0 \leq \sim r0$



r0

r1

r2

0x1000

0x0

0x1000

0x2000

0x0

0x2000

0x0

0x2004

0x0

0x2004

0x0

0x2008

0x0

0x2008

0x0

0x200c

0x0

0x200c

0x0

RTL 2

Memory Behaviour

$r0 \leq 0x5$ $r0 \leq r0 + 1$

$r1 \leq 0x1000$ $r2 \leq r2 + r0$

$r2 \leq m[r1]$

$r0 \leq \sim r0$



$r0$

$r1$

$r2$

$0x1000$

$0x0$

$0x1000$

$0x2000$

$0x0$

$0x2000$

$0x0$

$0x2004$

$0x0$

$0x2004$

$0x0$

$0x2008$

$0x0$

$0x2008$

$0x0$

$0x200c$

$0x0$

$0x200c$

$0x0$

RTL 2

Memory Behaviour

$r0 \leq 0x5$ $r0 \leq r0 + 1$

$r1 \leq 0x1000$ $r2 \leq r2 + r0$

$r2 \leq m[r1]$

$r0 \leq \sim r0$



$r0$

$r1$

$r2$

$0x1000$

$0x0$

$0x1000$

$0x2000$

$0x0$

$0x2000$

$0x0$

$0x2004$

$0x0$

$0x2004$

$0x0$

$0x2008$

$0x0$

$0x2008$

$0x0$

$0x200c$

$0x0$

$0x200c$

$0x0$

RTL 2

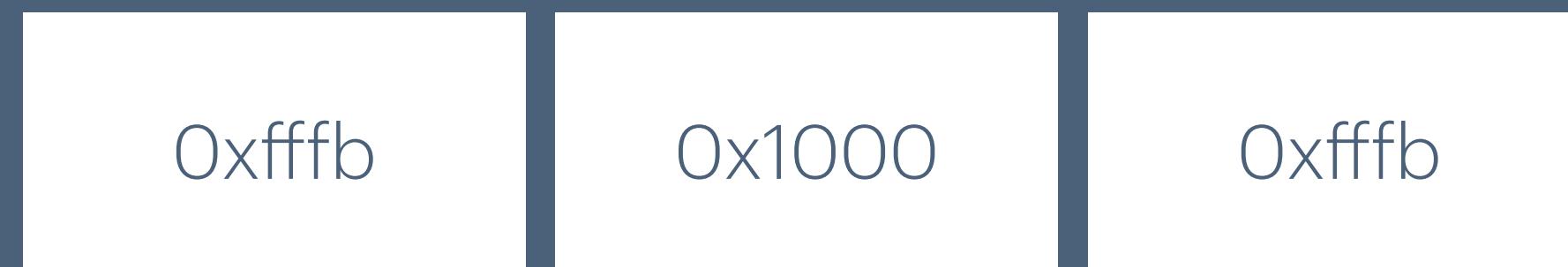
Memory Behaviour

$r0 \leq 0x5$ $r0 \leq r0 + 1$

$r1 \leq 0x1000$ $r2 \leq r2 + r0$

$r2 \leq m[r1]$ $m[r1] \leq r2$

$r0 \leq \sim r0$



0x1000 0x0

0x2000 0x0

0x2004 0x0

0x2008 0x0

0x200c 0x0

0x1000

0x2000 0x0

0x2004 0x0

0x2008 0x0

0x200c 0x0

RTL 2

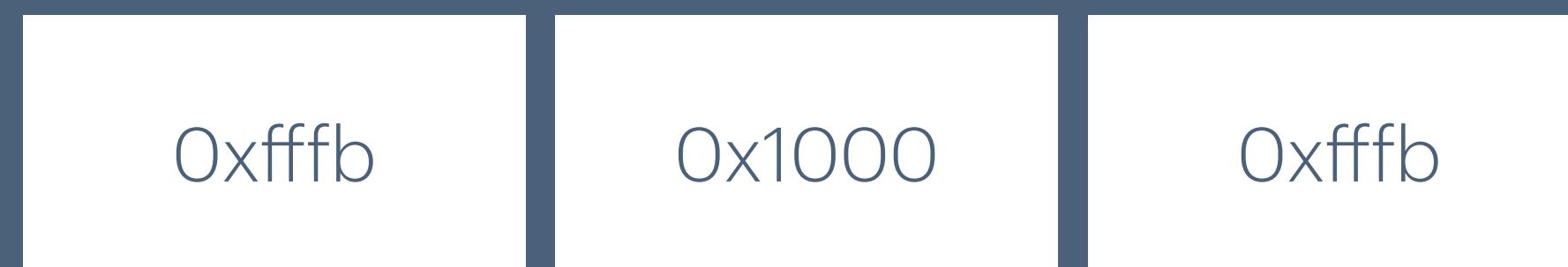
Memory Behaviour

$r0 \leq 0x5$ $r0 \leq r0 + 1$

$r1 \leq 0x1000$ $r2 \leq r2 + r0$

$r2 \leq m[r1]$ $m[r1] \leq r2$

$r0 \leq \sim r0$



0x1000
0x0

0x1000
0xffffb

0x2000
0x0

0x2000
0x0

0x2004
0x0

0x2004
0x0

0x2008
0x0

0x2008
0x0

0x200c
0x0

0x200c
0x0

RTL 2

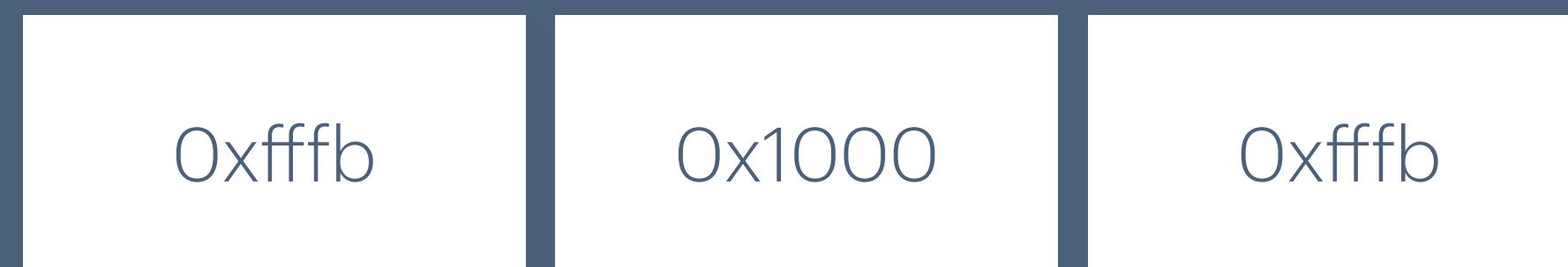
Memory Behaviour

$r0 \leq 0x5$ $r0 \leq r0 + 1$

$r1 \leq 0x1000$ $r2 \leq r2 + r0$

$r2 \leq m[r1]$ $m[r1] \leq r2$

$r0 \leq \sim r0$



0x1000

0x0

0x1000

(dec) -5

0x2000

0x0

0x2000

0x0

0x2004

0x0

0x2004

0x0

0x2008

0x0

0x2008

0x0

0x200c

0x0

0x200c

0x0

Lab Code

For attendance

HQLA

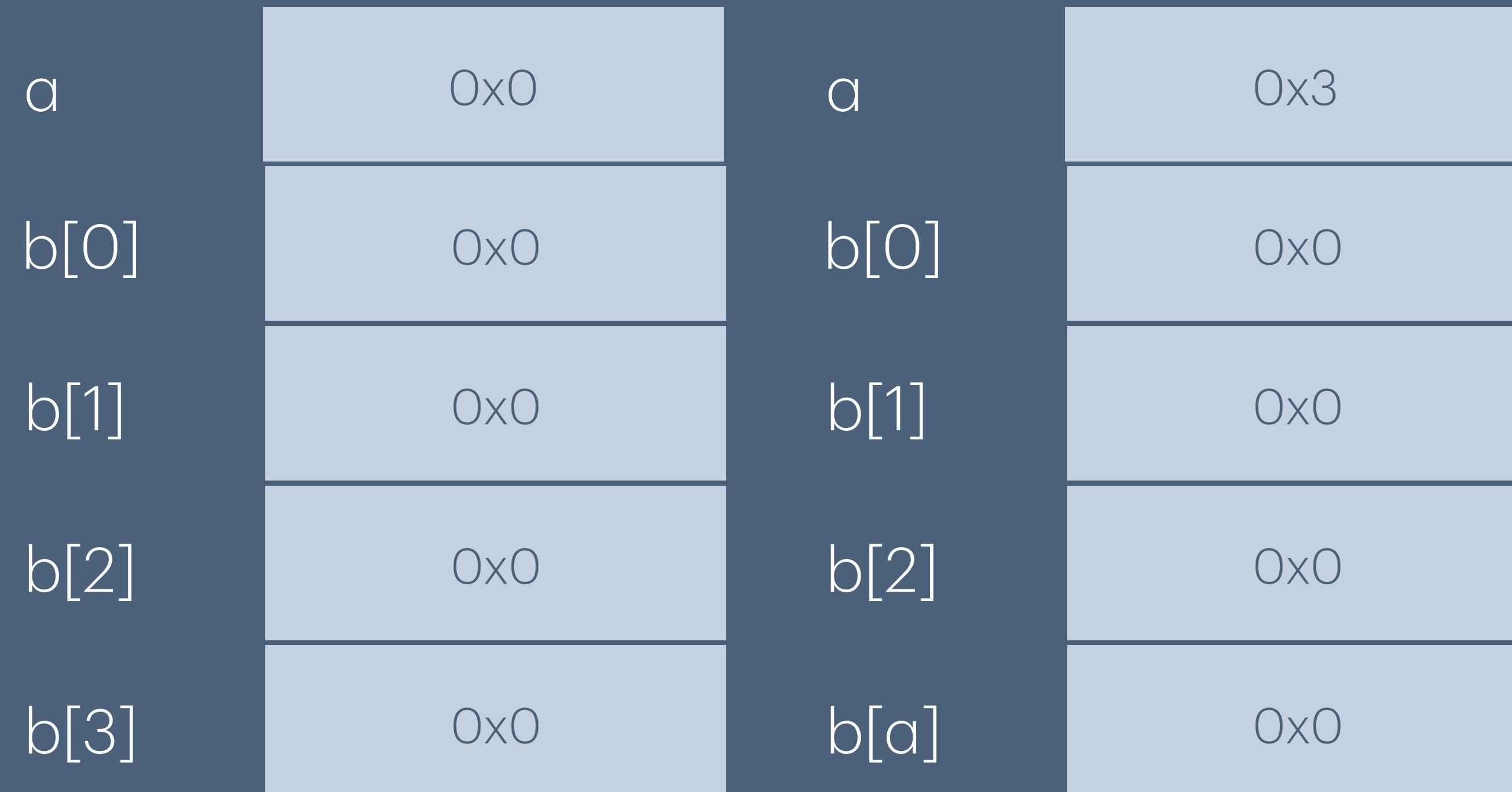
C to SM213 Assembly

Memory layout

```
int a;
int b[10];
```

That is, you can assume that

```
a: .long 0
b: .long 0
    .long 0
    .long 0
    .long 0 # enough to
```



C to SM2132 Assembly

Translate $b[a] = a + b[a];$

- Get value of a
 - Location of a
- Get value of $b[a]$
 - b starts, and the value of a; need indexed load
- Add them together
- Put it into $b[a]$
 - b starts and value of a; need indexed store

C to SM2132 Assembly

Get value of a

- I need the value of a

ld \$a, r1 # where a is

ld (r1), r1 # value of a

C to SM2132 Assembly

Get value of b[a]

- I need where b starts, and the value of a

ld \$b, r0 # where b starts

ld \$a, r1 # where a is

ld (r1), r1 # value of a

ld (r0, r1, 4), r2 # r2 = b[a]

C to SM2132 Assembly

Add together

- overwrite r2 because the only use for the **value of** $b[a]$ isn't used next

`ld $b, r0 # where b starts`

`ld $a, r1 # where a is`

`ld (r1), r1 # value of a`

`ld (r0, r1, 4), r2 # r2 = b[a]`

`add r1, r2 # r2 = b[a] + a`

C to SM2132 Assembly

Store into b[a]

- $r2 = b[a] + a$, put into $m[b + a * 4]$

ld \$b, r0 # where b starts

ld \$a, r1 # where a is

ld (r1), r1 # value of a

ld (r0, r1, 4), r2 # $r2 = b[a]$

add r1, r2 # $r2 = b[a] + a$

st r2, (r0, r1, 4)