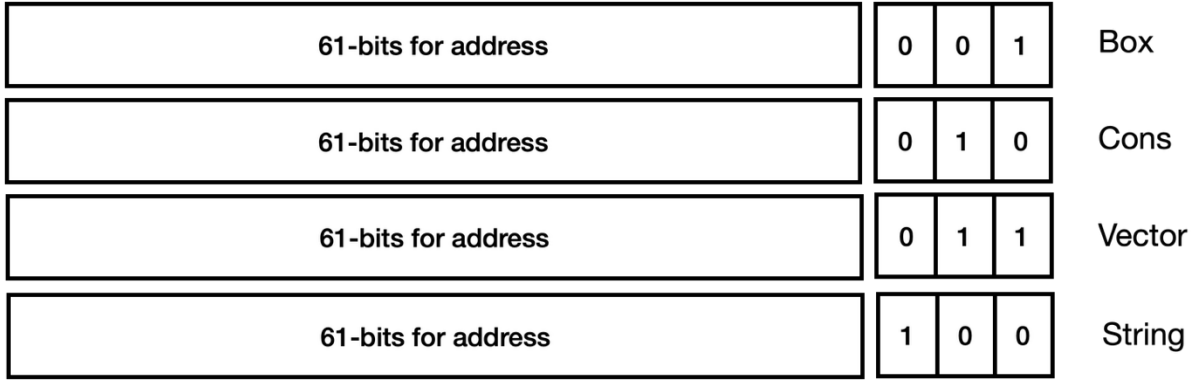
CMSC 430: Introduction to Compilers

Hoax: vectors and strings

Hoax

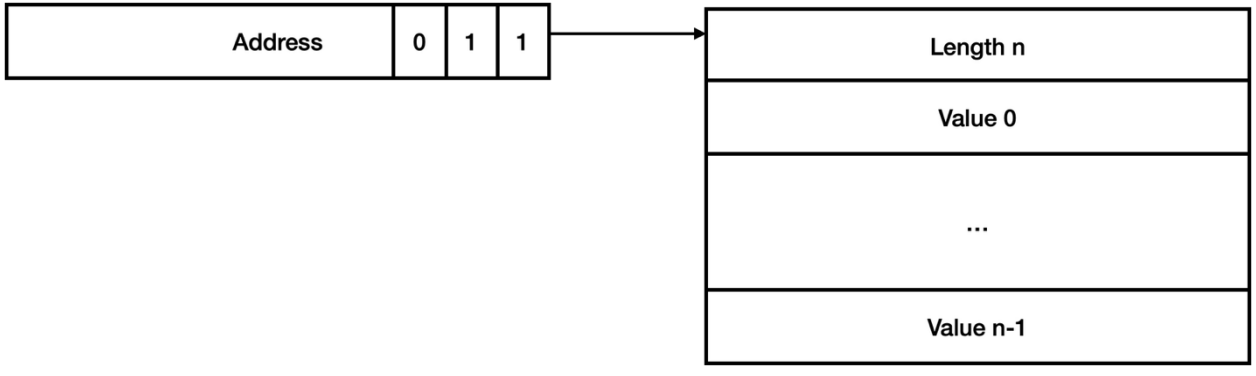
- Hoax adds vectors and strings.
  - vectors: heterogenous arrays
  - strings: homogenous arrays
- New operations and values:
  - **`vector? vector-length`**
  - **`string? string-length`**
  - **`make-vector vector-ref`**
  - **`make-string string-ref`**
  - **`vector-set!`**

Hoax: Encoding the Pointer Values



How to represent vectors?

- Vector: arrays that carry their size.
  - Every access checks bounds
  - Vectors are mutable



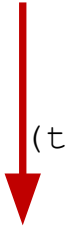
Example: (make-vector 5 1)

Registers		
rax	H | 0x3	(vector ptr)
rbx	H + 48	(next free)
r8	0	
r9	40	

Heap	
Offset	**Bits**
H + 0	80
H + 8	16
H + 16	16
H + 24	16
H + 32	16
H + 40	16
H + 48	—

Example: (make-vector 0 1)

Registers	
rax	&empty + 3 (tagged empty-vector ptr)
rbx	H (unchanged – no allocation)
r8	0 (n=0, fixnum tag 0b0000)



rax = &empty+3
(type-vect tag = 0b011)

Data Section		
Address	**Label**	**Value (64-bit)**
&empty	empty	0x0000000000000000

make-vector

value is in rax
Length is in stack

```
(let ((nonzero (gensym 'nz))
(loop (gensym 'loop))
(theend (gensym 'theend)))
(seq (Pop r8)
(assert-natural r8)
(Cmp r8 0) ; special case for
length = 0
(Jne nonzero)
(Lea rax (Mem 'empty type-vect))
(Jmp theend)

(Label nonzero)
(Mov (Mem rbx 0) r8) ; write length
(Sar r8 1) ; convert to bytes
(Mov r9 r8) ; heap adjustment
```

```
; start initialization
(Label loop)
(Mov (Mem rbx r8) rax)
(Sub r8 8)
(Cmp r8 0)
(Jne loop)
; end initialization


(Mov rax rbx)
(Xor rax type-vect)
(Add rbx r9); elements
(Add rbx 8) ; length
(Label theend)))]
```

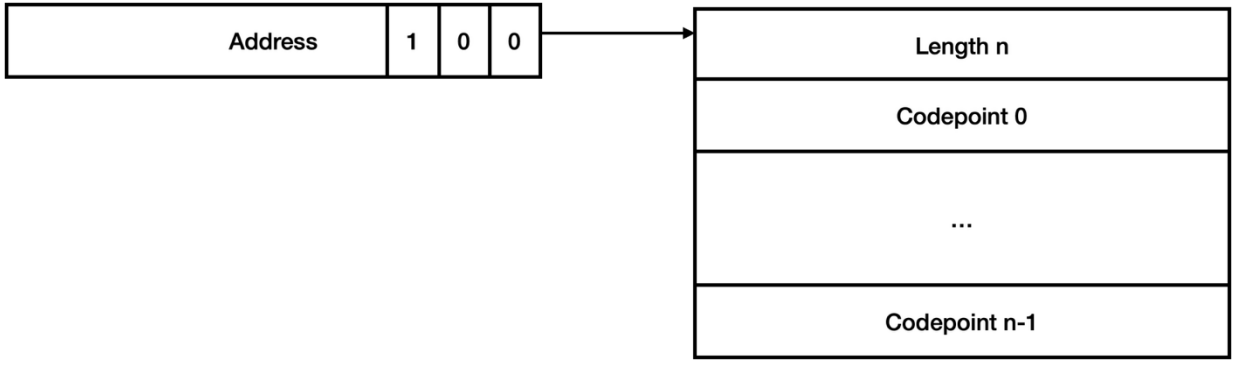
vector-ref

rax: index
Stack: vector pointer

```
(assert-vector r8)
(assert-natural rax)
(Mov r9 (Mem r8 (- type-vect)))
(Cmp rax r9)
(Jge 'err)
(Sar rax 1); rax: index * 8
(Mov rax (Mem r8 rax (- 8 type-vect))))]
```

How to Represent Strings?

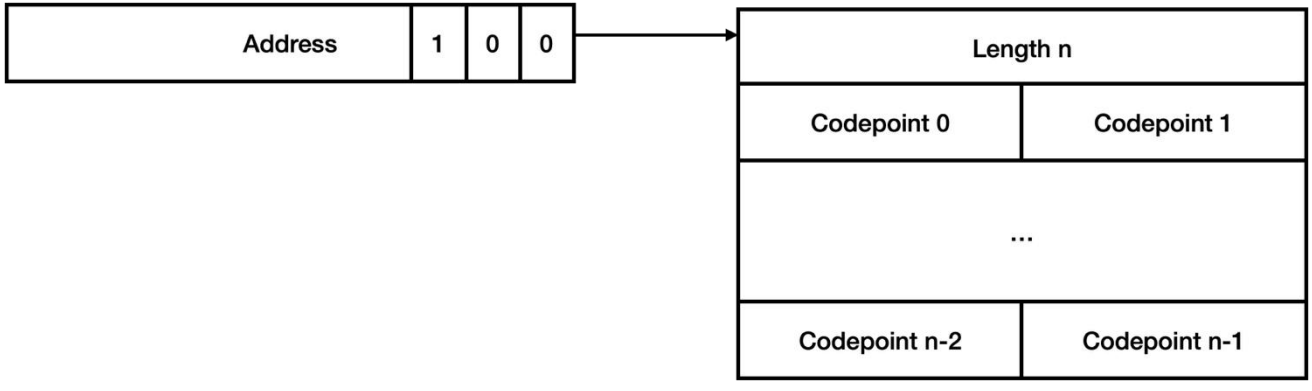
- Strings are codepoint arrays that carry their size
  - Every access checks bounds
  - Strings are immutable



Codepoints are only 21-bits wide so this is wasteful.

How to Represent Strings?

- Each code point is 21-bits. We can store two codepoints in one 64-bit.



Memory Layout after: (make-string 5 #\a) → "aaaaa"

Inputs: rax=3112 (tagged #\a, codepoint=3112>>5=97), r8=80 (tagged int 5, len=80>>4=5)

Registers (after)	
rax	H | 4 ← tagged string ptr (XOR with 4 = string tag)
rbx	H + 32 ← new heap top (advanced past allocation)
r8	0 ← loop counter exhausted

rax = H|4
(bit2=tag=str)

Heap	
H + 0	80 = (5 << 4) ← tagged int, string length = 5
H + 8	0x00000061 = 97 ← codepoint[0] = 'a'
H + 12	0x00000061 = 97 ← codepoint[1] = 'a'
H + 16	0x00000061 = 97 ← codepoint[2] = 'a'
H + 20	0x00000061 = 97 ← codepoint[3] = 'a'
H + 24	0x00000061 = 97 ← codepoint[4] = 'a'
H + 28	0x00000000 ← alignment padding (odd len → 4 pad bytes)
H + 32	← *rbx (free heap starts here)*