Lab_Interning: String Interning Implementation

J. Nelson Amaral

Motivation

298 no-space characters

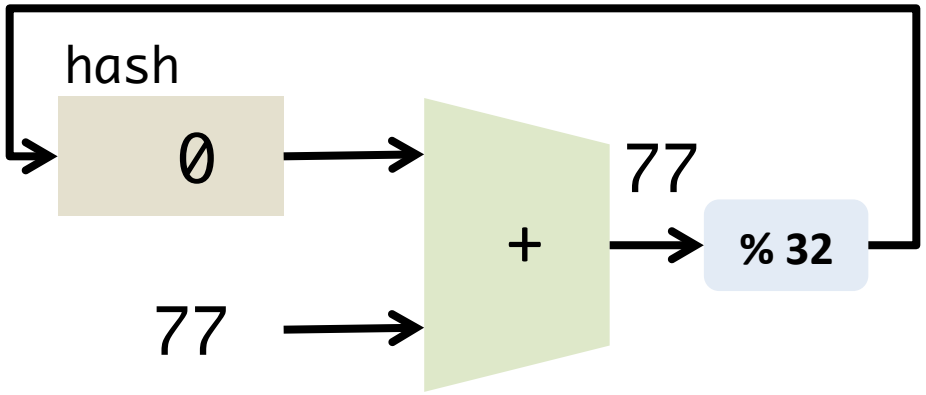
```
MilisecondsPerSecond = 1000;
SecondsPerMinute = 60;
MinutesPerHour = 60;
SecondsPerHour = SecondsPerMinute * MinutesPerHour;

def ConvertHourMinuteSecondToSeconds(
              ElapsedHous,
              ElapsedMinutes,
              ElapsedSeconds) :
  return [ElapsedHours * SecondsPerHour
          + ElapsedMinutes * SecondsPerMinute
          + ElapsedSeconds]

def HowManyElapsedHours(ElapsedSeconds) :
  return [ElapsedSeconds / SecondsPerHour]

def HowManyElapsedMinutes(ElapsedSeconds) :
  return [ElapsedSeconds / SecondsPerMinute
          - (ElapsedSeconds / SecondsPerHour) * MinutesPerHour]
```

```
id00 = 1000;
id01 = 60;
id02 = 60;
id03 = id01 * id02;

def id04(
              id05,
              id06,
              id07) :
  return [id05 * id03
          + id06 * id01
          + id07]

def id08(id07) :
  return [id07 / id03]

def id08(id07) :
  return [id07 / id01
          - (id07 / id03) * id02]
```

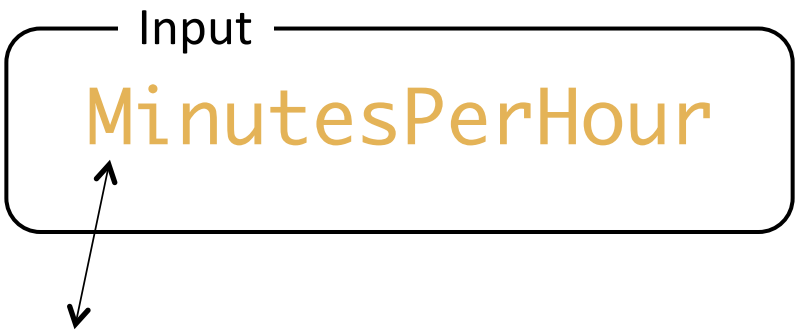
199 no-space characters

33% reduction in size

Hashing Algorithm

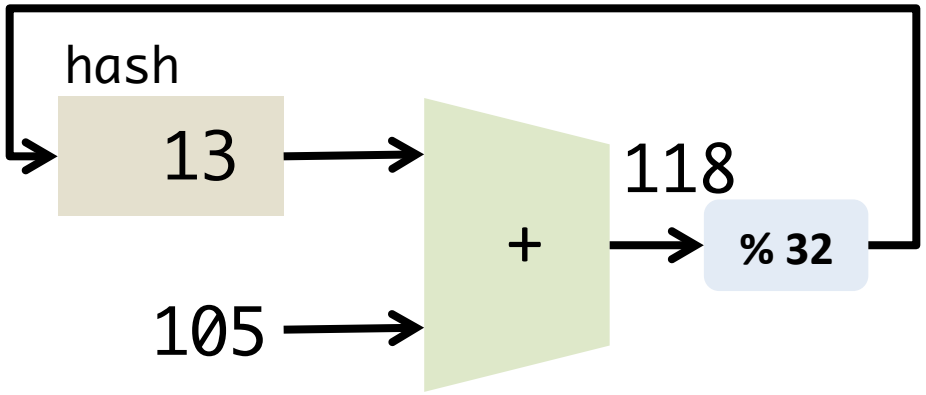



```
data[]
hash = 0
for d in data
        do
        hash = (hash + d) mod n
return hash
```

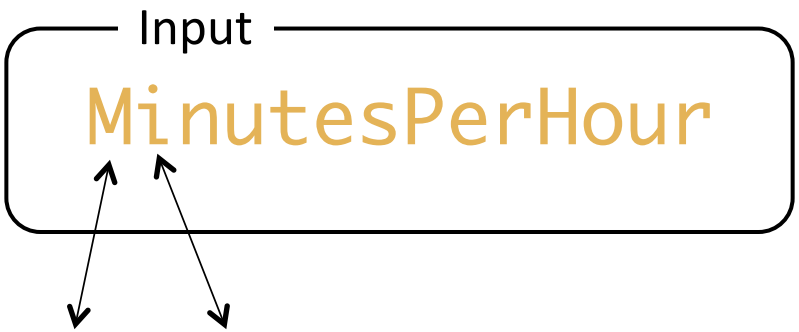


ASCII Codes: 77

Hashing Algorithm

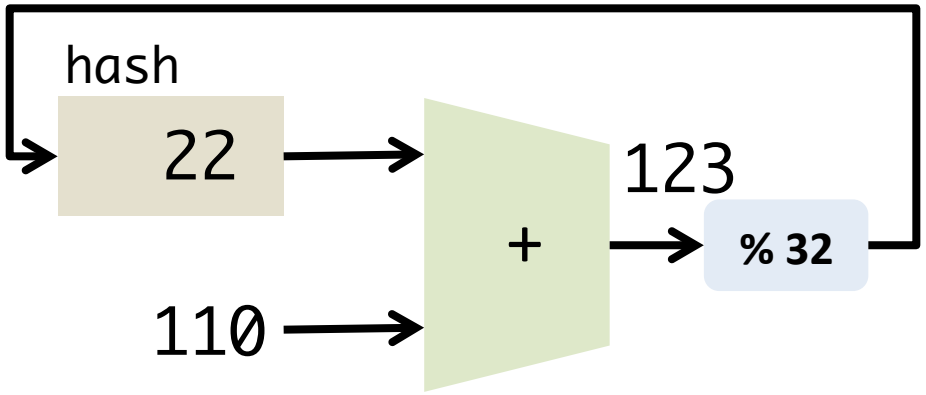



```
data[]
hash = 0
for d in data
        do
        hash = (hash + d) mod n
return hash
```

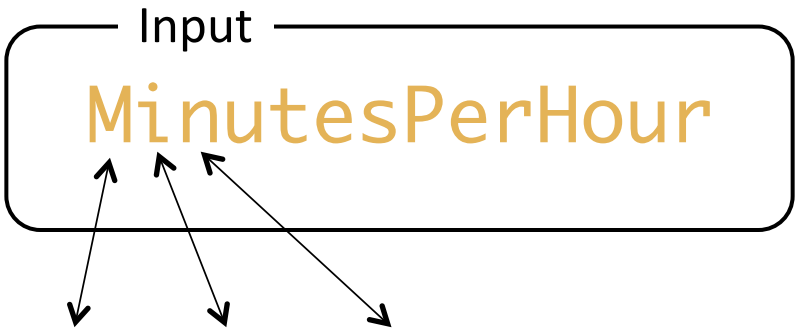


ASCII Codes: 77 105

Hashing Algorithm

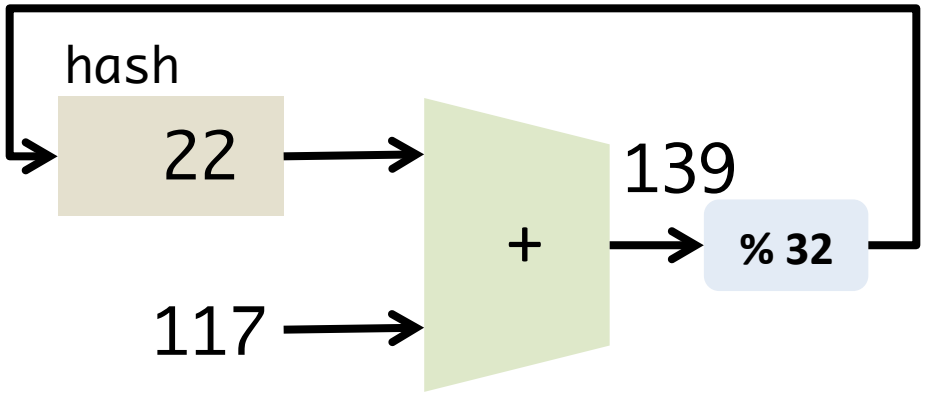



```
data[]
hash = 0
for d in data
        do
        hash = (hash + d) mod n
return hash
```

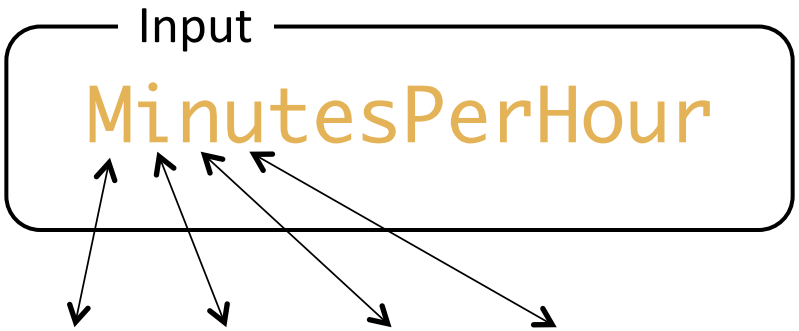


ASCII Codes: 77 105 110

Hashing Algorithm

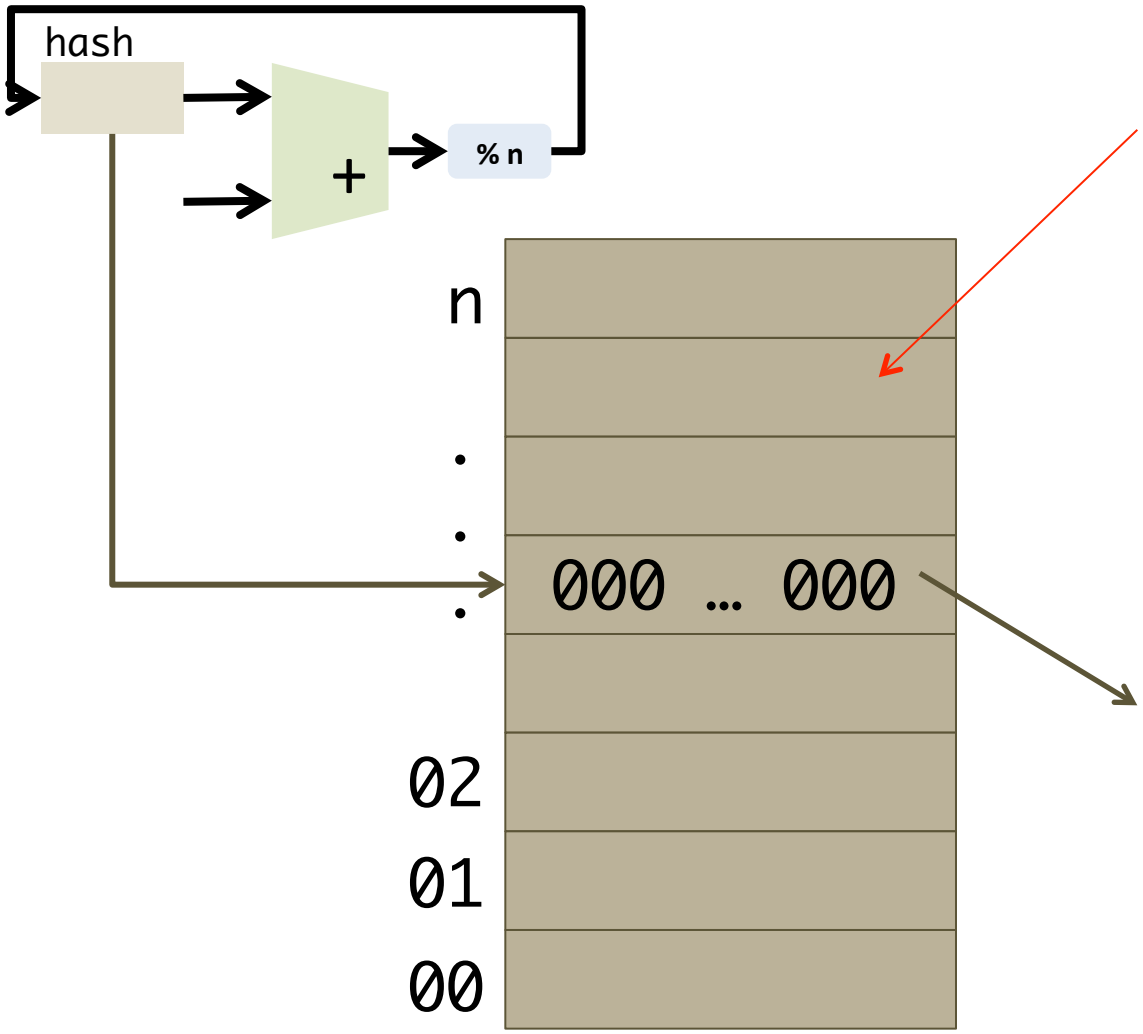



```
data[]
hash = 0
for d in data
        do
        hash = (hash + d) mod n
return hash
```



ASCII Codes: 77 105 110 117 …

String Interning Table

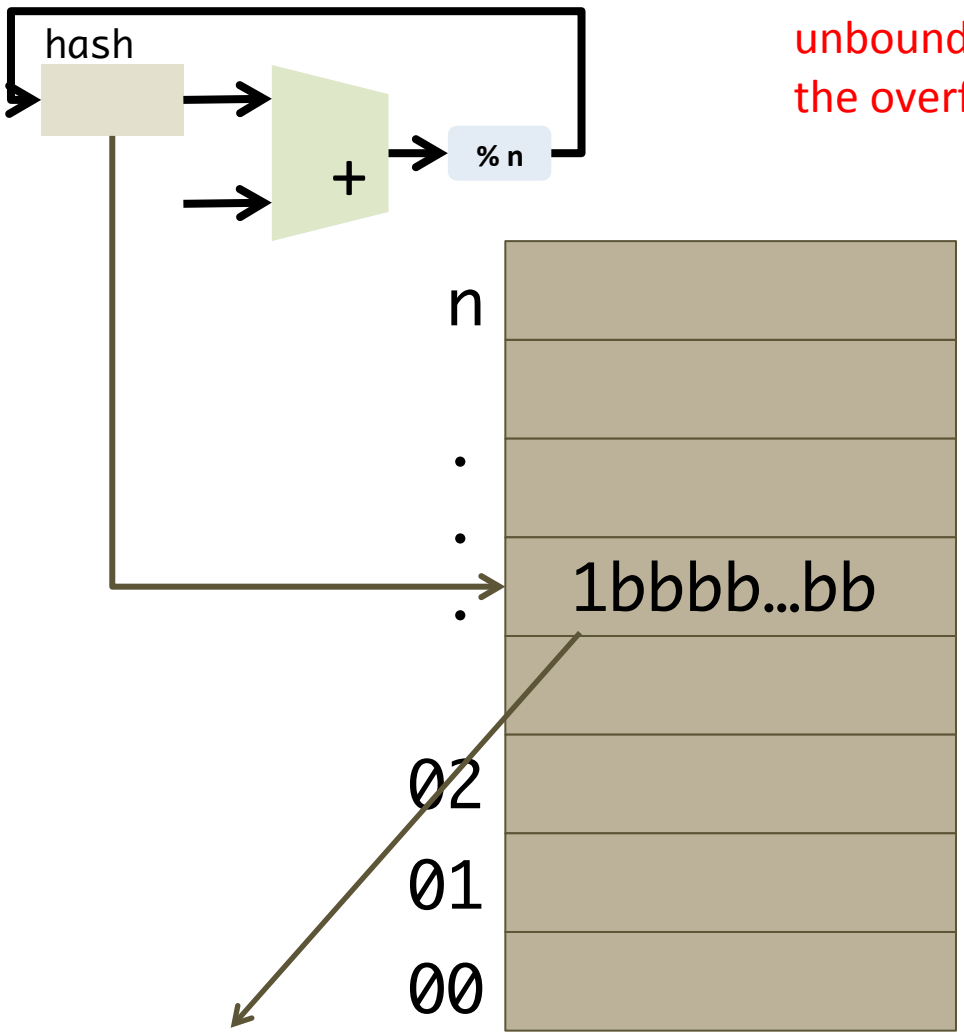


Each one of these entries in the table is called a *bucket*

- **Create** immutable copy
- **Place** address of immutable copy into the table entry.
- **Return** unique ID

String Interning Table

In this slides we will refer to the unbounded data structure used for the overflowing of a bucket as a ***tank***



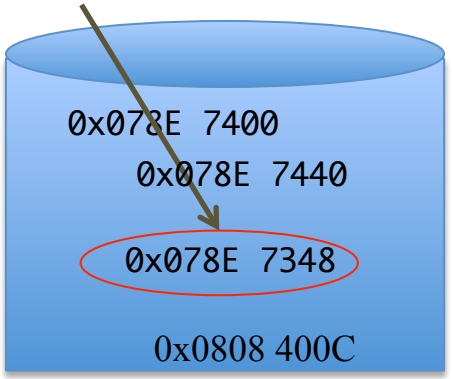
Search tank for string.

If not found:

- **create** immutable copy
- **insert** string into tank.
- **return** unique id

If found:

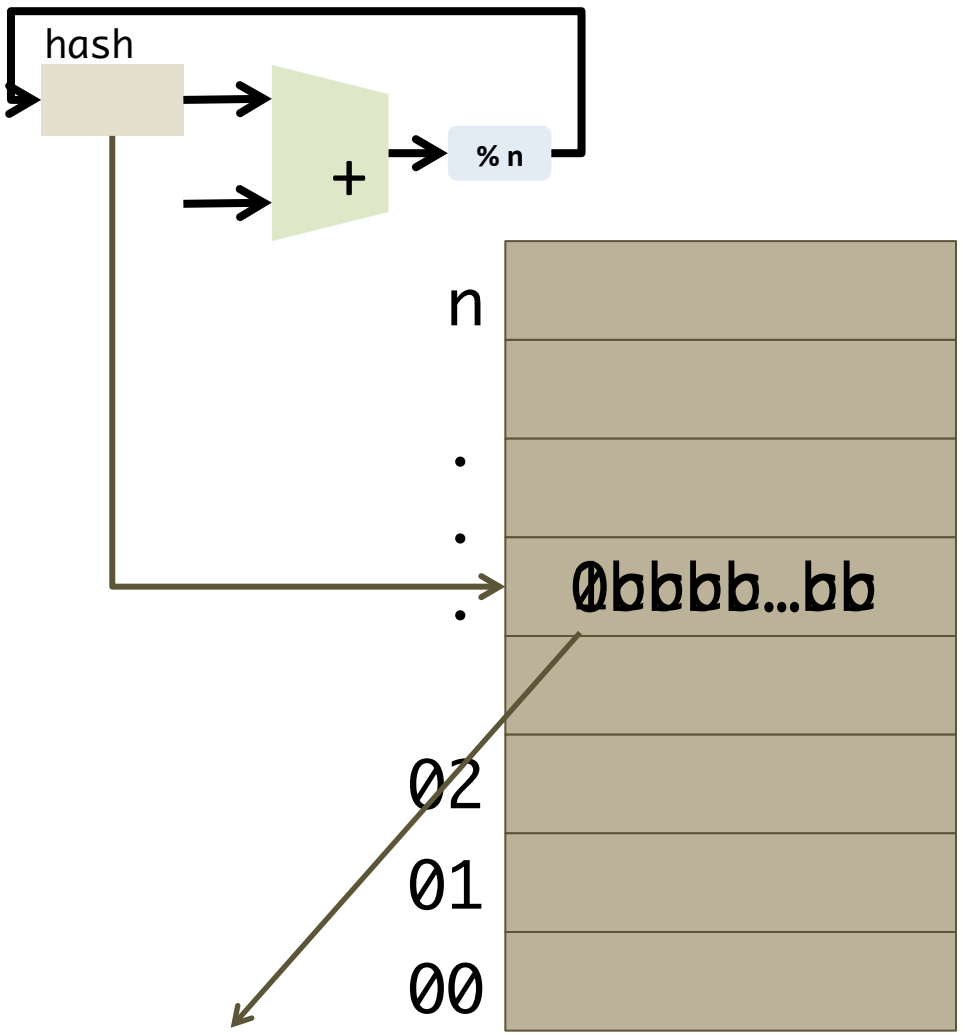
- **return** unique ID.



0bbbb…bb is the address of an interned string

〈string〉

String Interning Table



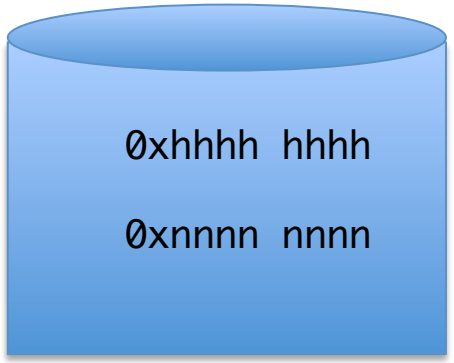
Compare input with interned string.

If it matches:

- **return** unique id

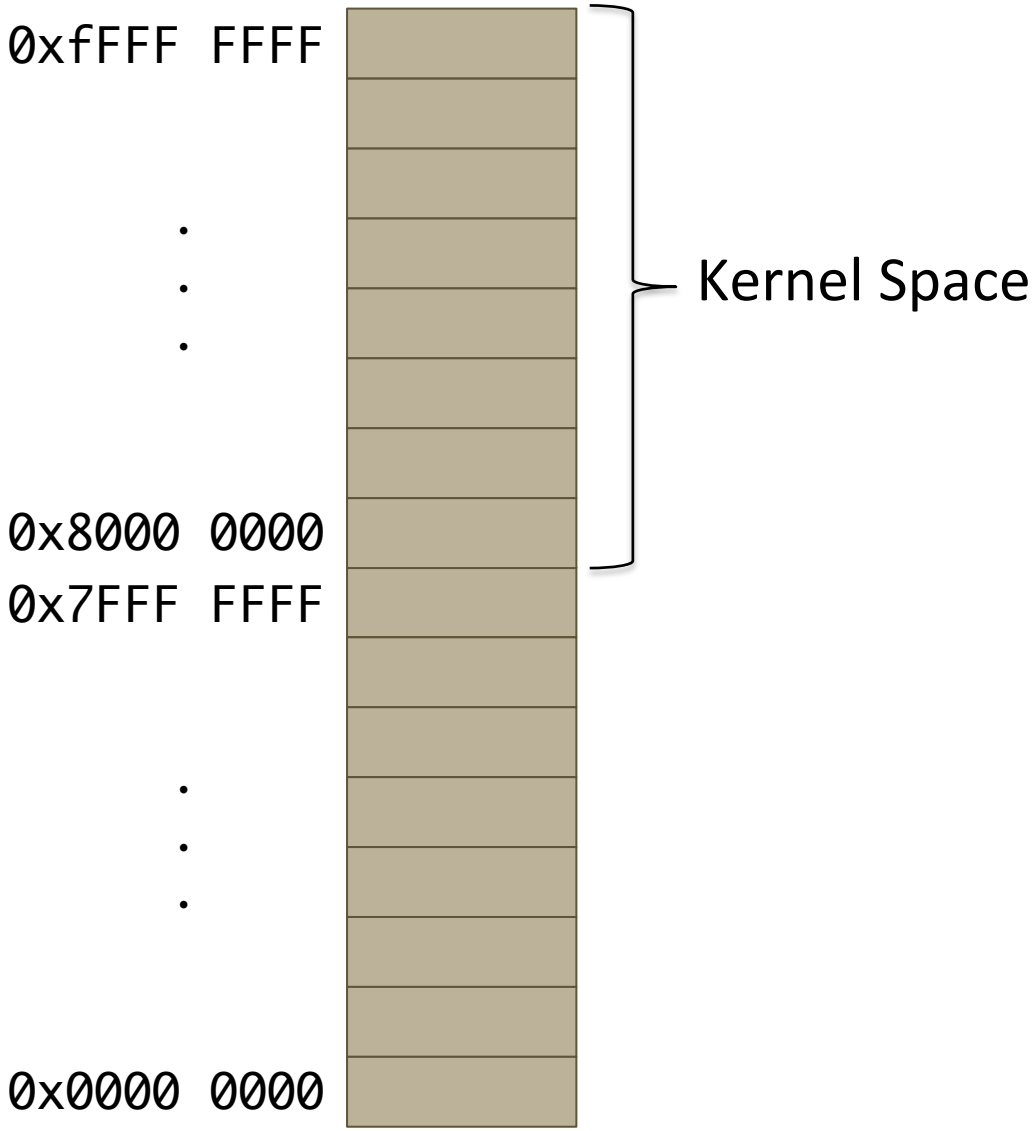
If it does not match:

- **create** a new tank.
- **add** interned string address
- **create** immutable copy
- **add** copy address
- **replace** bucket content
- **return** unique ID



0bbbb…bb is the address of an interned string

Why this works?



internString

Parameter:

$a0: address of a string to be interned

Return Value:

$v0: unique ID for the string

Strings with same value must always return the same ID independent of their memory address.

Must create immutable copy if string was not interned before.

getInternedString

Parameter:

`$a0`: unique interned string ID

Return Value:

`$v0`: string address — if string was interned before.

`$v0`: `0` — if string was not interned before.

internFile

Parameter:

$a0: pointer to a file in mutable memory

Return Value:

$v0: address of list of interned strings

$v1: number of identifiers in the list

- strings are separated by one or more spaces or line feed characters
- end of file is signaled by an End of Transmission (EOT) character
- must make immutable copies of each string.

Guarantee

Your implementation will be presented with
at most 128 different strings.