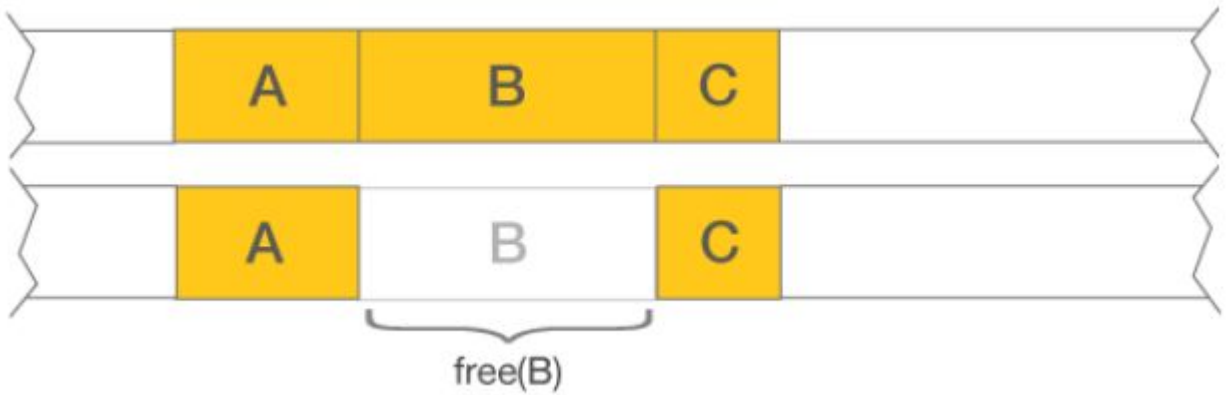
Как GC освобождает память

Сафин Рустам

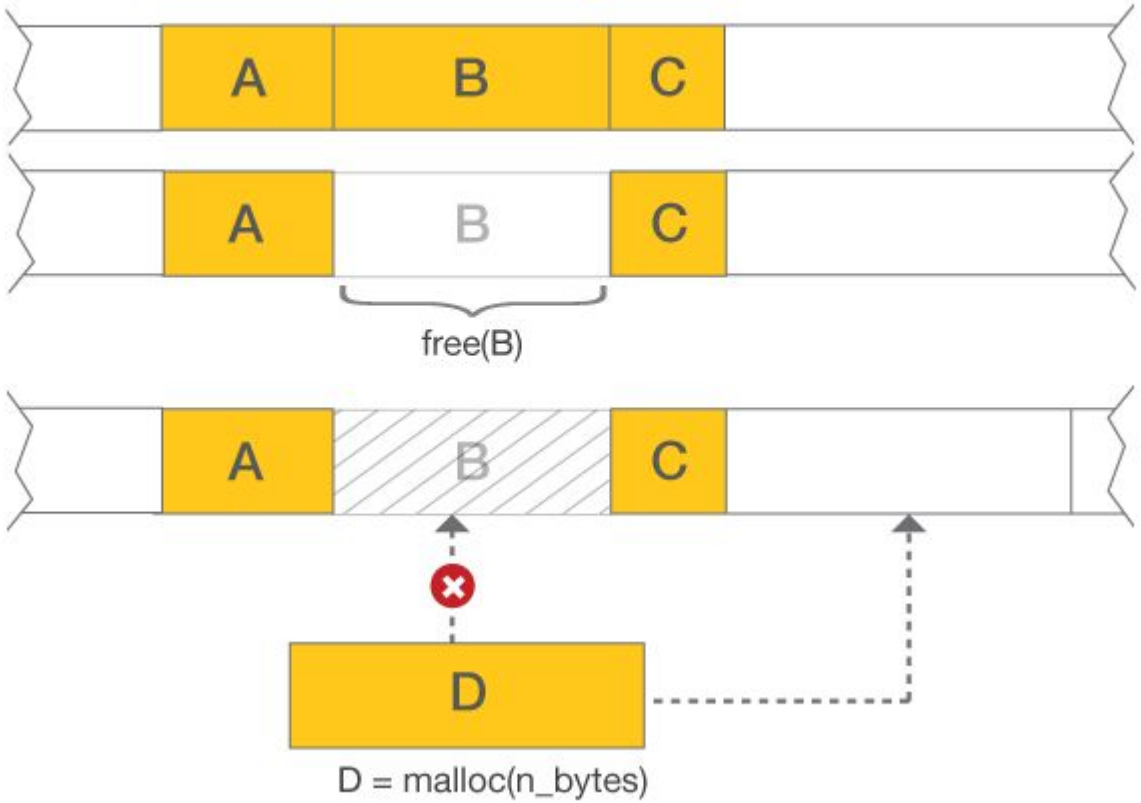
DIRECTUM

Фрагментация

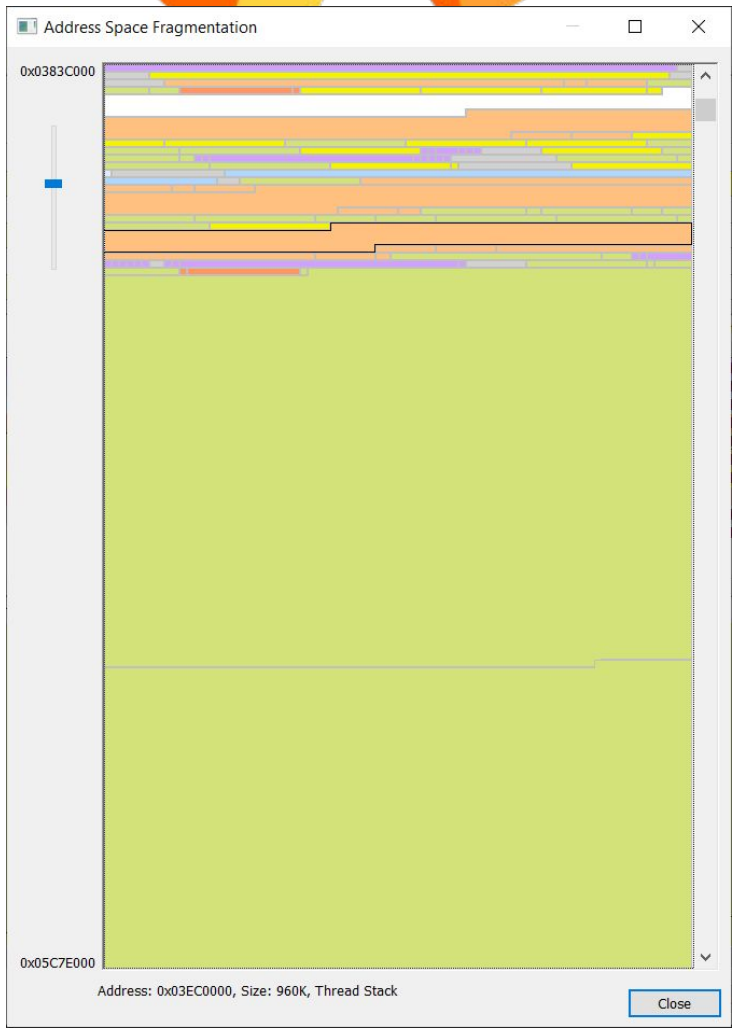
Фрагментация



Фрагментация



Память процесса



Thread stack

Gen2

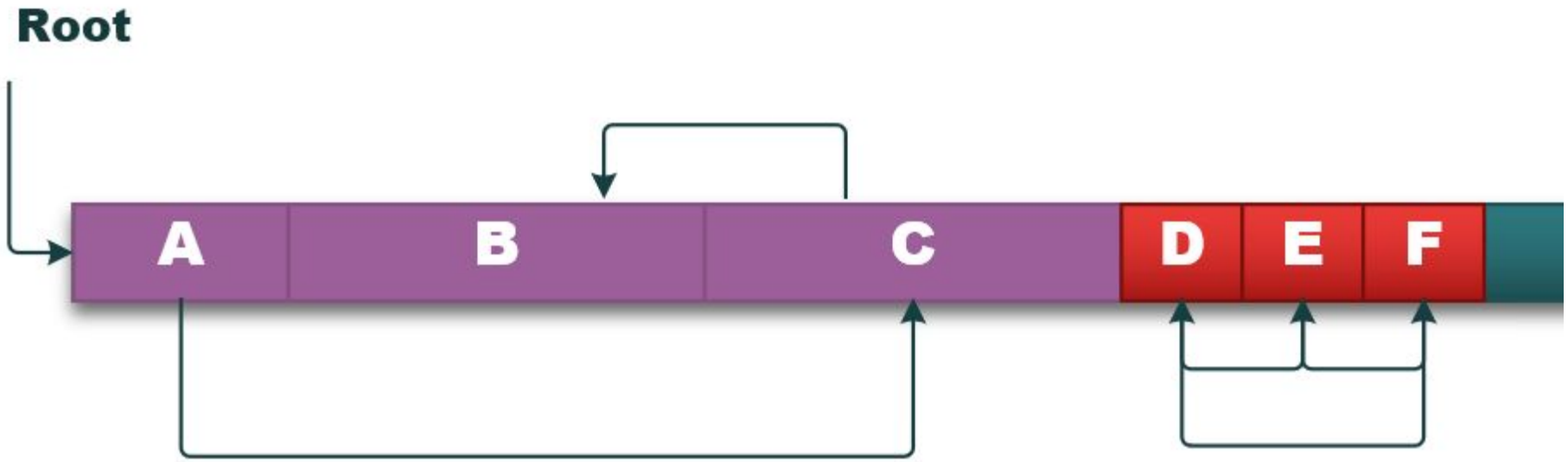
Gen1

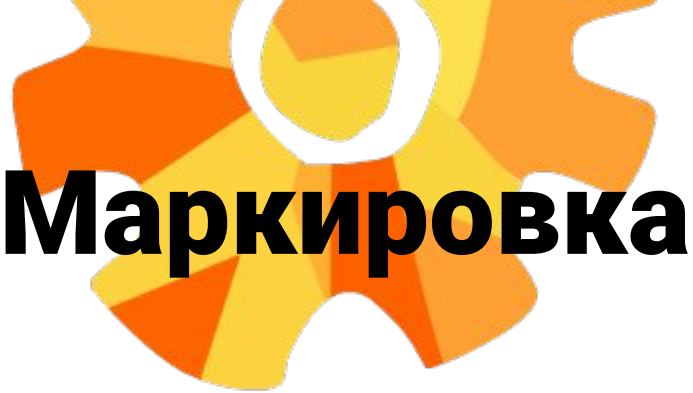
Этапы

```
GarbageCollectGeneration()
 {
     SuspendEE();
     garbage_collect();
     RestartEE();
 }

 garbage_collect()
 {
     generation_to_condemn();
     gc1();
 }
```

```
gc1()
 {
     mark_phase();
     plan_phase();
 }

 plan_phase()
 {
     if (compact)
     {
         relocate_phase();
         compact_phase();
     }
     else
         make_sweep();
 }
```

Маркировка

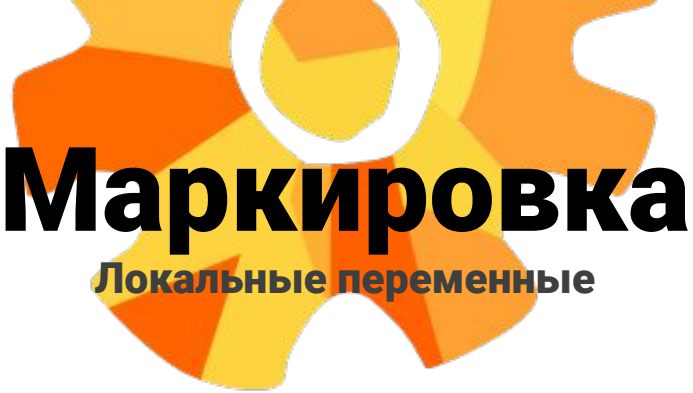




Маркировка

Корни

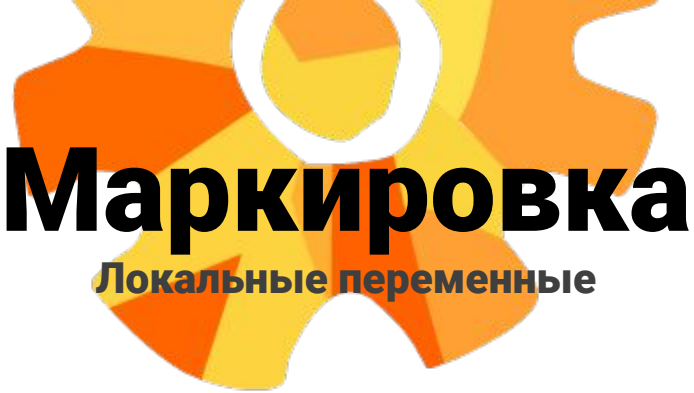
- Стек локальных переменных
- Таблица финализаторов
- Таблица Handle
- Карточные столы



Маркировка

Локальные переменные

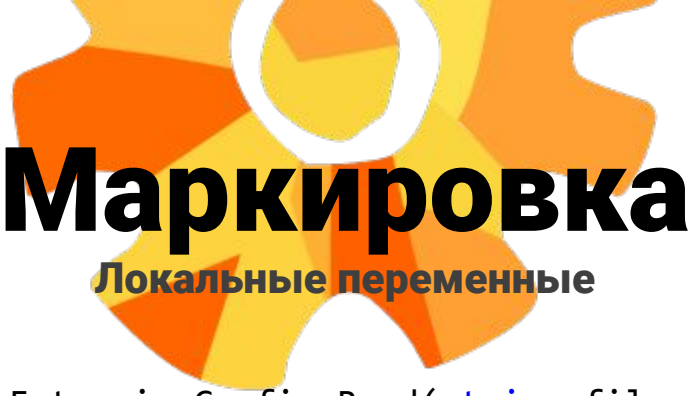
```
public static ExtensionConfig Read(string filename) {
  var config = new ExtensionConfig();
  if (!File.Exists(filename))
      return config;
  var doc = XDocument.Load(filename, LoadOptions.None);
  var root = doc.Root;
  if (root.Name == XML_ROOT_NAME)
      Read(root, config);
  return config;
}
```



Маркировка

Локальные переменные

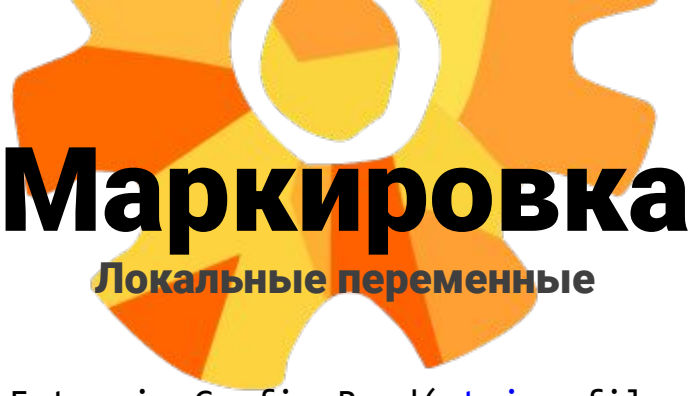
```
public static ExtensionConfig Read(string filename) {
  var config = new ExtensionConfig();                    //
  if (!File.Exists(filename))                            // config
      return config;                                     // config
  var doc = XDocument.Load(filename, LoadOptions.None);  // config, doc
  var root = doc.Root;                                   // config, doc, root
  if (root.Name == XML_ROOT_NAME)                        // config, root
      Read(root, config);                                // config, root
  return config;                                         // config
}
```



Маркировка

Локальные переменные

```
public static ExtensionConfig Read(string filename) {
  var config = new ExtensionConfig();                          //
  if (!File.Exists(filename))                                  // config
      return config;                                           // config
  var doc = XDocument.Load(filename, LoadOptions.None);        // config, doc
  var root = doc.Root;                                         // config, doc, root
  if (root.Name == XML_ROOT_NAME)                              // config, root
      Read(root, config);                                      // config, root

  fixed (IntPtr* ptr = this.accessor.DbBinding) {              // config, ptr
      // ..                                                    // config, ptr
  }                                                            // config

  return config;                                               // config
}
```



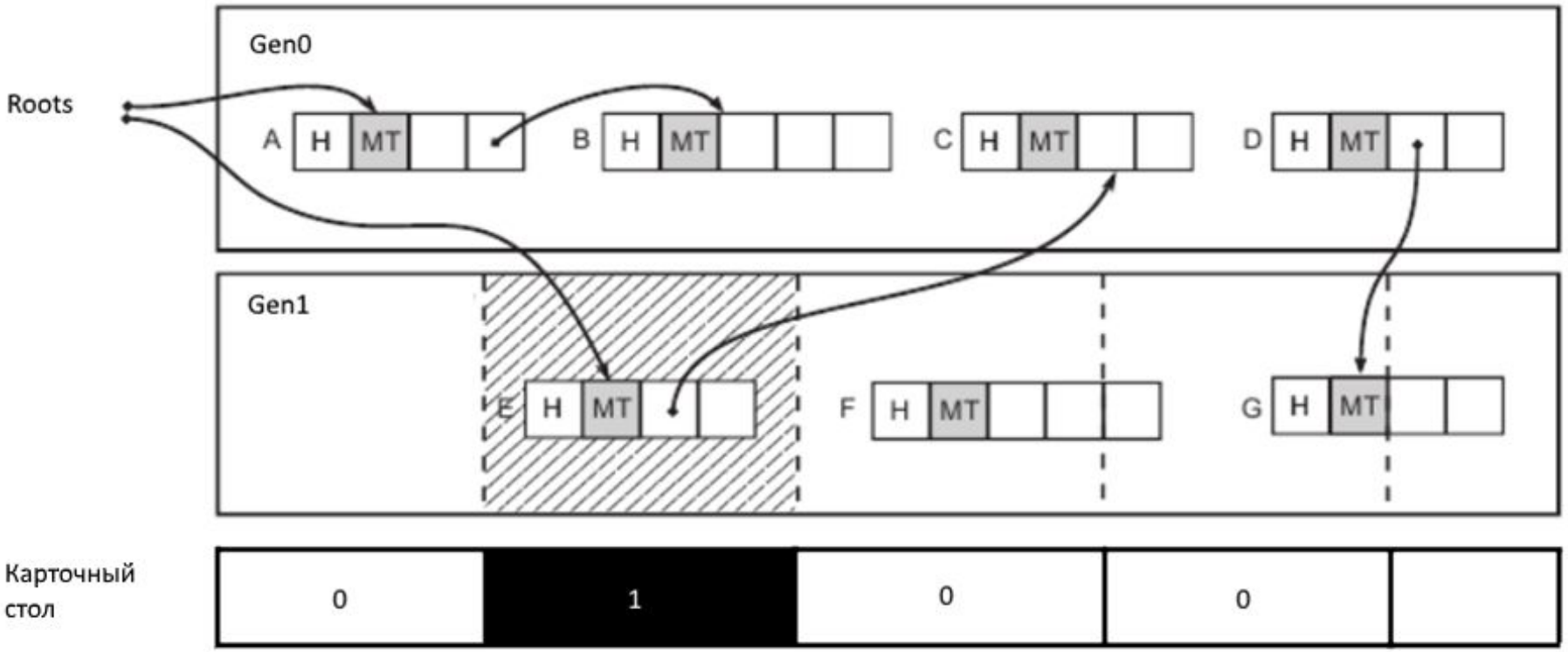
Маркировка

Локальные переменные

```
public static ExtensionConfig Read(string filename) {
  var config = new ExtensionConfig();                          //
  if (!File.Exists(filename))                                  // rax
      return config;                                           // rax
  var doc = XDocument.Load(filename, LoadOptions.None);        // rax, rbx
  var root = doc.Root;                                         // rax, rbx, rcx
  if (root.Name == XML_ROOT_NAME)                              // rax, rbx
      Read(root, config);                                      // rax, rbx

  fixed (IntPtr* ptr = this.accessor.DbBinding) {              // rax, rbx [pinned]
      // ..                                                    // rax, rbx [pinned]
  }                                                            // rax

  return config;                                               // rax
}
```

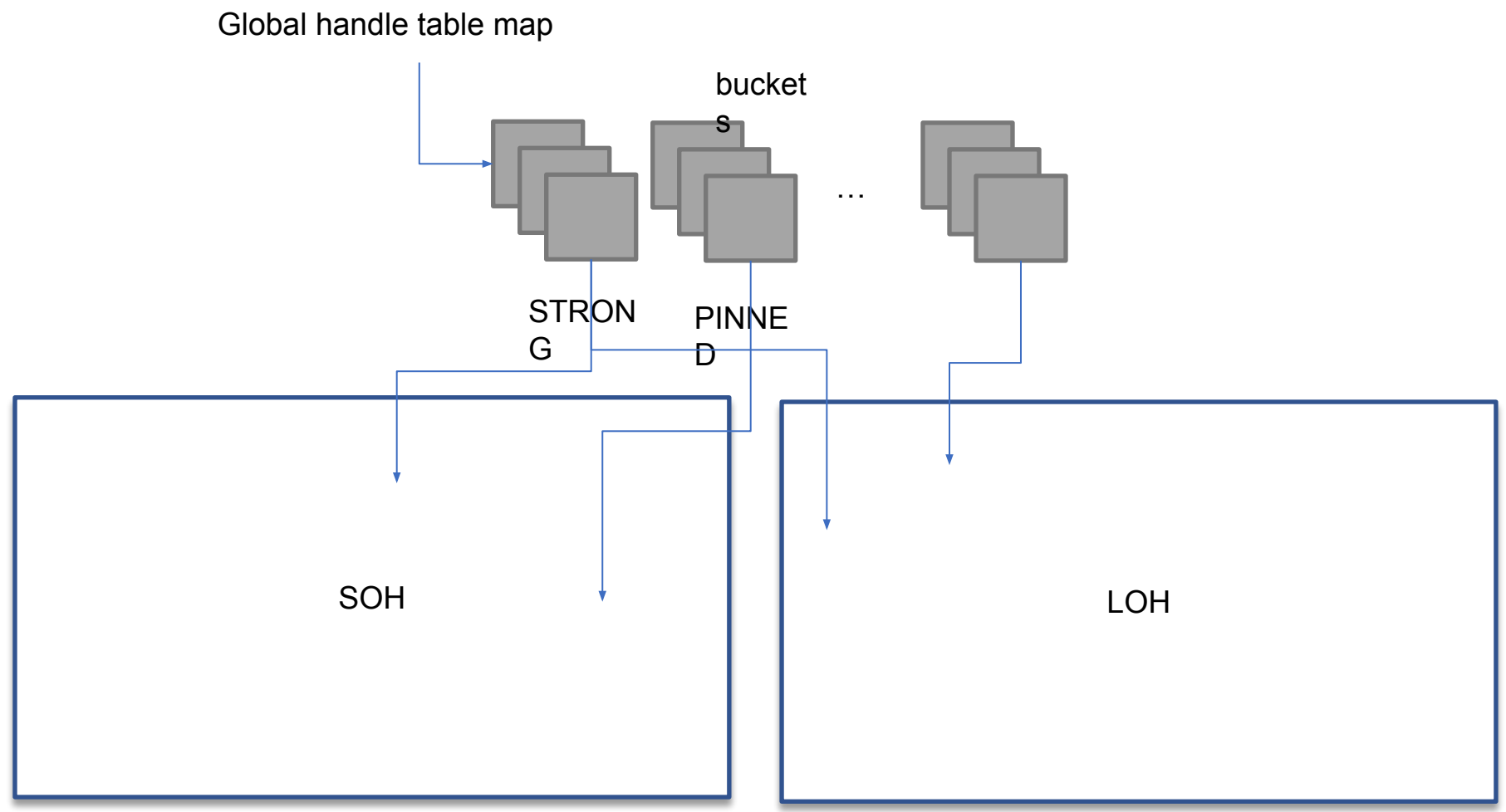
Маркировка

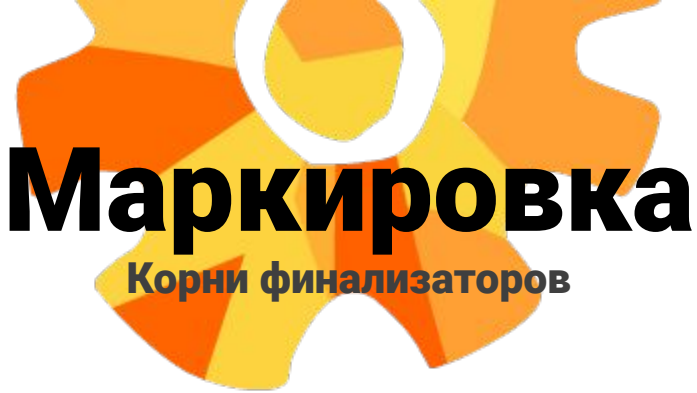
Карточные столы



Маркировка

Таблица Handle

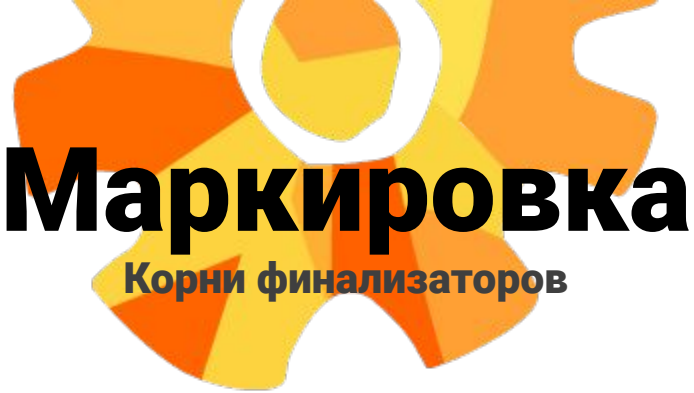




Маркировка

Корни финализаторов

```
public class Foo
{
    public void DownloadFromUrl(string url)
    {
        using (var client = new HttpClient())
        {
            this.result = client.Get(url);
        }
    }

    ~Foo()
    {
        Console.WriteLine("In finalizer");
    }
}
```



Маркировка

Корни финализаторов

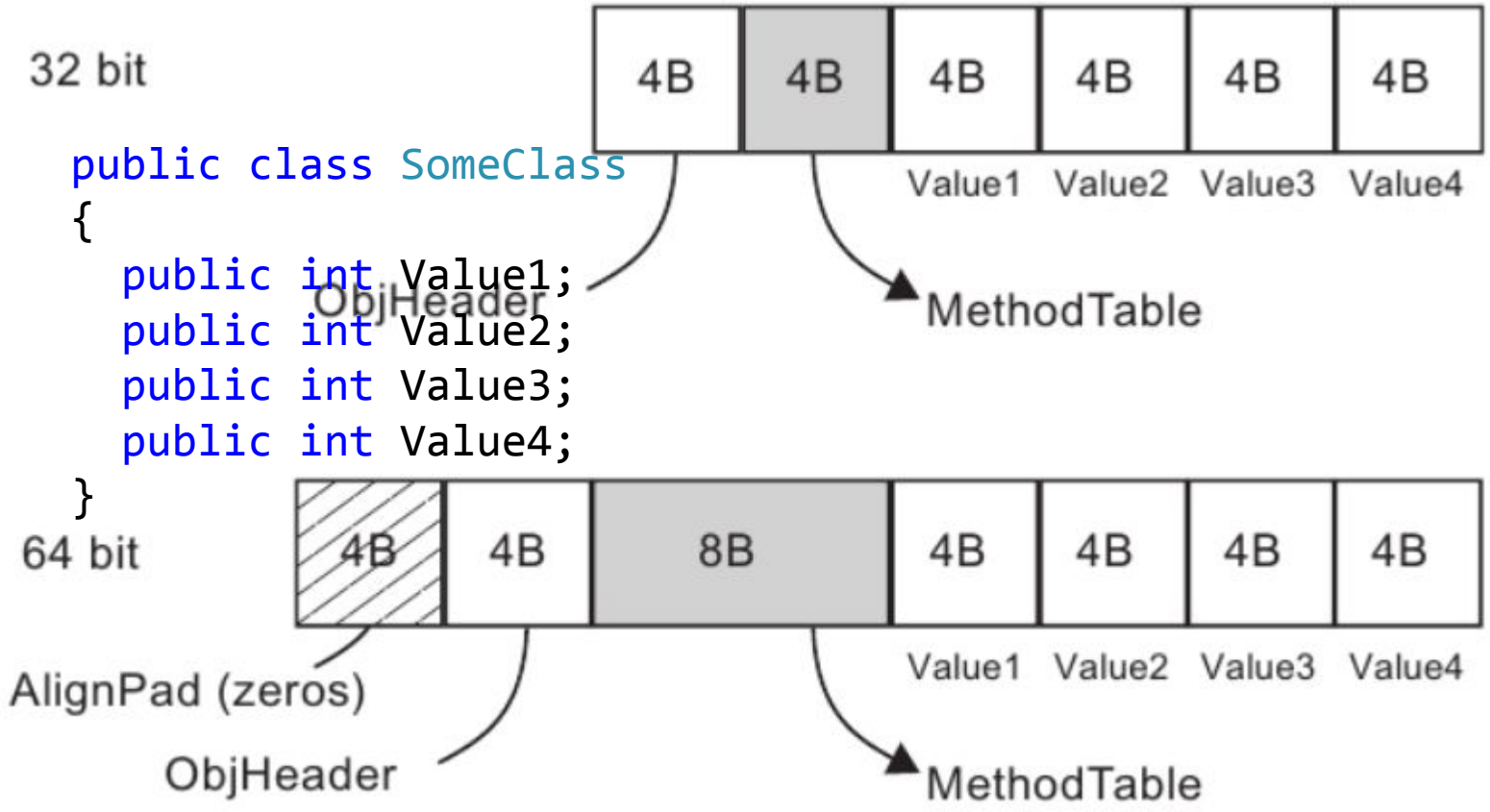
```
{
        void DownloadFromUrl(Foo this, string url)
    {
        using (var client = new HttpClient())
        {
            this.result = client.Get(url)
        }
    }

    ~Foo(Foo this)
    {
        Console.WriteLine("In finalizer");
    }
}
```

Флаг маркировки

```
public class SomeClass
{
  public int Value1;
  public int Value2;
  public int Value3;
  public int Value4;
}
```

Флаг маркировки

```
public class SomeClass
{
  public int Value1;
  public int Value2;
  public int Value3;
  public int Value4;
}
```



Флаг маркировки

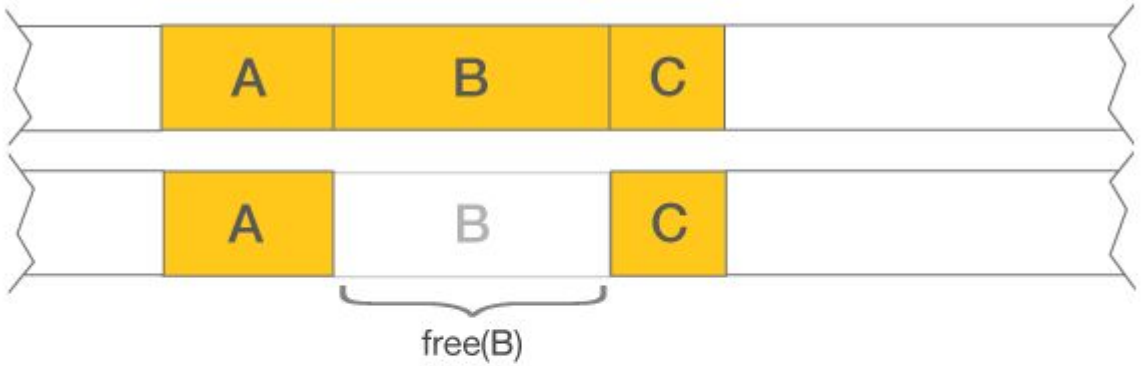
```
#ifdef _TARGET_64BIT_
#define OBJHEADER_SIZE      (sizeof(DWORD) + sizeof(DWORD))
#else
#define OBJHEADER_SIZE       sizeof(DWORD)
#endif
```

```
#define set_marked(obj) header(obj)->SetMarked()
```

```
BOOL gc_heap::gc_mark1 (uint8_t* obj)
{
    BOOL marked = !marked (obj);
    set_marked (o);
    return marked;
}
```

```
void set_marked()
{
  RawSetMethodTable(RawGetMethodTable()) | 0x1);
}
```

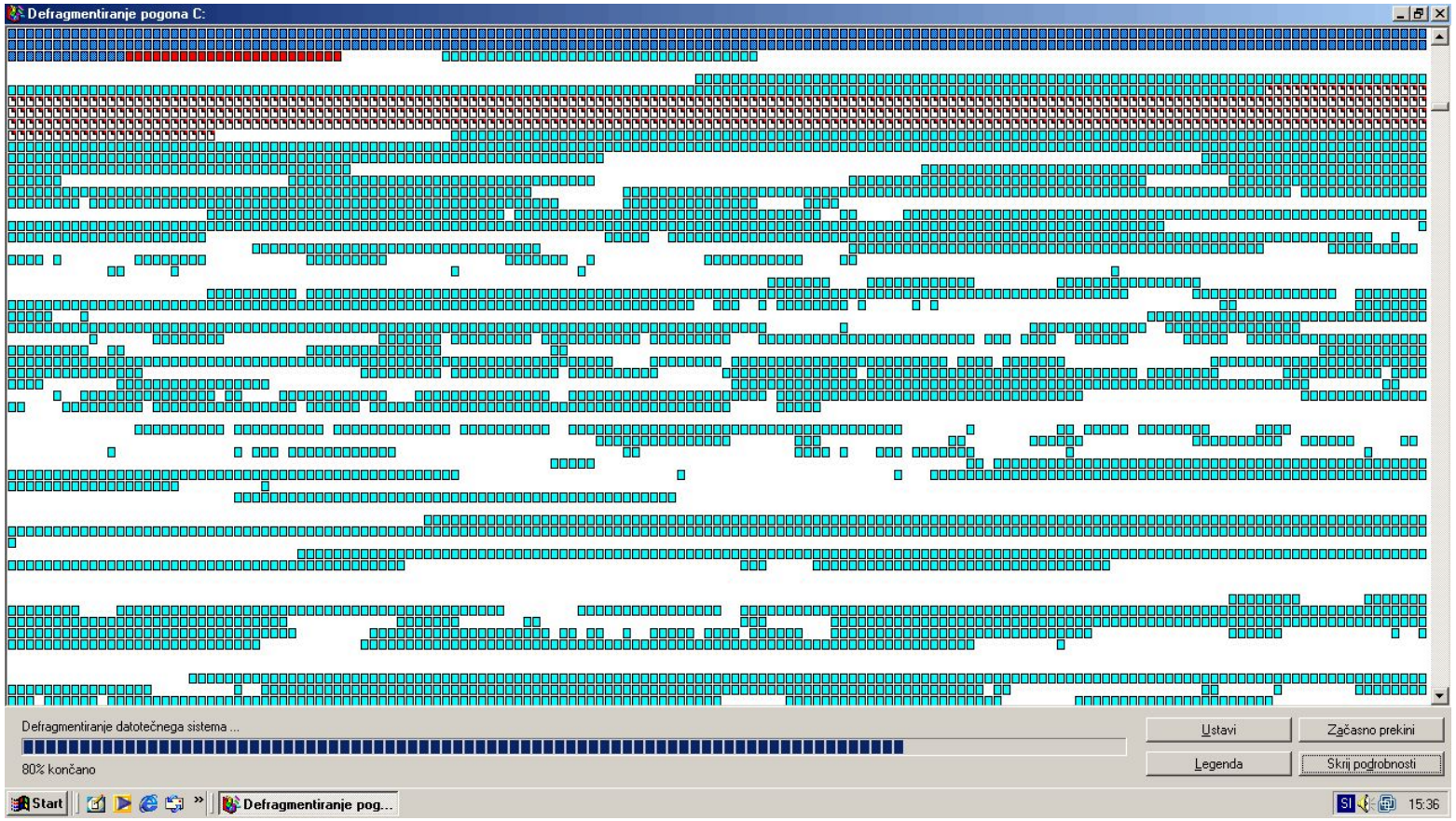
Sweep



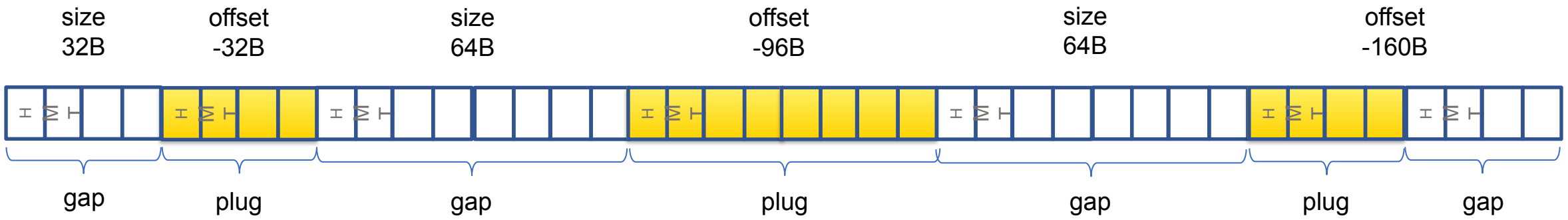
Sweep

```
void sweep(Root* root) {
    Object** object = &root->next;
    while (*object) {
        if (!(*object)->marked) {
            /* Объект не маркирован, можем освободить память. */
            Object* unreached = *object;
            *object = unreached->next;
            free(unreached);
        } else {
            /* Объект маркирован. Снимем отметку для следующего GC и перейдем к следующему.
*/
            (*object)->marked = 0;
            object = &(*object)->next;
        }
    }
}
```

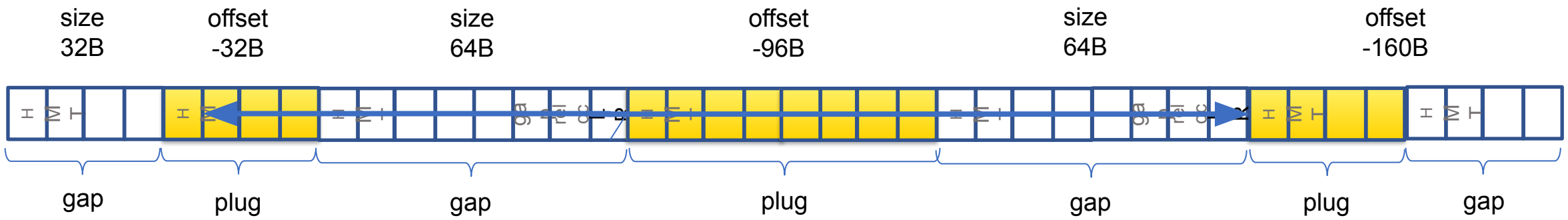
Compact


Defragmentiranje pogona C:
Defragmentiranje datotečnega sistema ...
80% končano
Ustavi
Začasno prekini
Legenda
Skrij podrobnosti
Start
Defragmentiranje pog...
SI
15:36

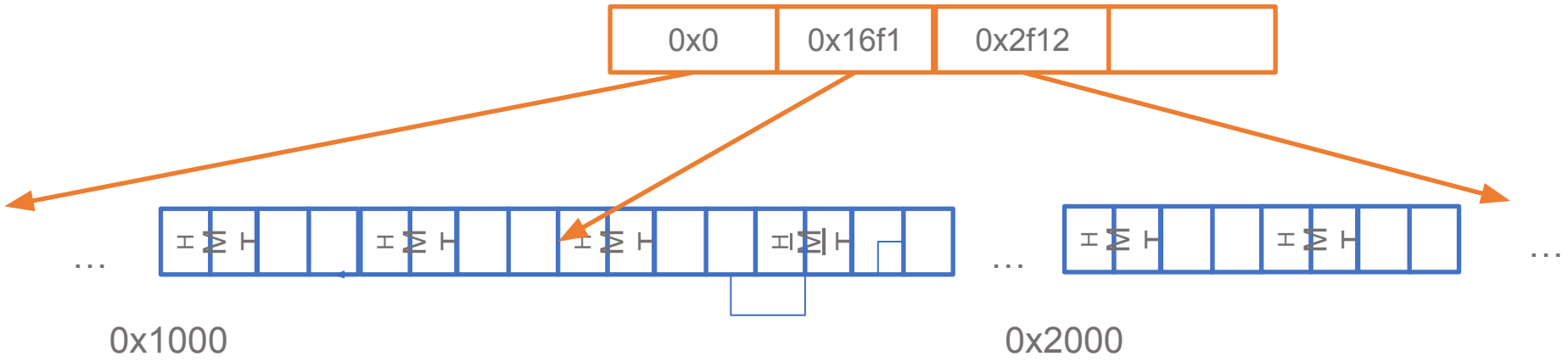
Compact


size
32B
offset
-32B
size
64B
offset
-96B
size
64B
offset
-160B
gap
plug
gap
plug
gap
plug
gap

Compact


size
32B
offset
-32B
size
64B
offset
-96B
size
64B
offset
-160B
gap
plug
gap
plug
gap
plug
gap

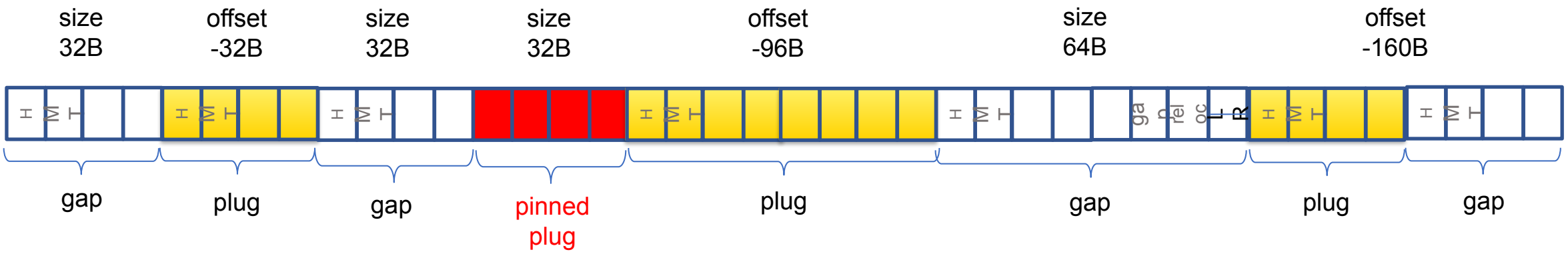
Brick table



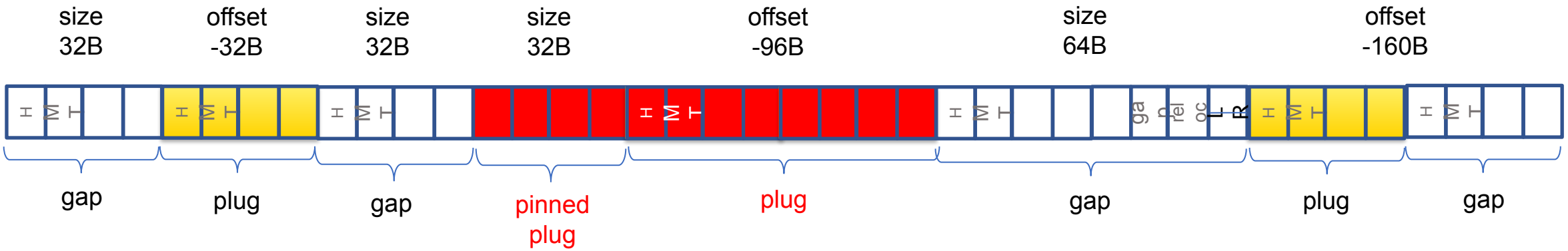
Compact

```
internal unsafe struct MyBuffer
{
      public fixed char fixedBuffer[128];
}


fixed (char* charPtr = myBuffer.fixedBuffer)
{
      *charPtr = 'A';
}
```

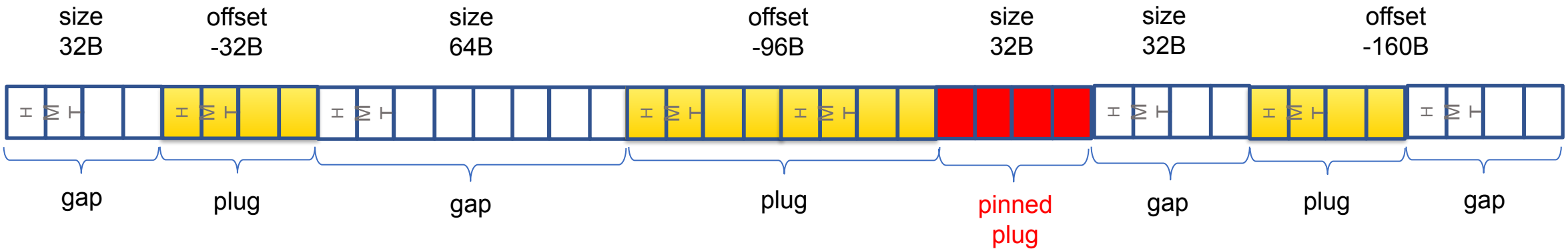
Compact


size
32B
offset
-32B
size
32B
size
32B
offset
-96B
size
64B
offset
-160B
gap
plug
gap
pinned
plug
plug
gap
plug
gap

Compact


size
32B
offset
-32B
size
32B
size
32B
offset
-96B
size
64B
offset
-160B
gap
plug
gap
pinned
plug
plug
gap
plug
gap

Compact


size
32B
offset
-32B
size
64B
offset
-96B
size
32B
size
32B
offset
-160B
gap
plug
gap
plug
pinned
plug
gap
plug
gap

Сафин Рустам

Разработчик
DIRECTUM в г. Уфа

safin_rf@directum.ru