Operativni sustavi (I067)
3. vježbe

Množenje

```
// Returns x*y, where x, y ≥ 0
multiply(x, y):
  sum = 0
  shiftedX = x
  for  i = 0 ... w - 1   do
      if  ((i'th bit of y) == 1)
          sum = sum + shiftedX
      shiftedX = shiftedX * 2
  return sum
```

x :	...	0	0	0	**1**	**1**	**0**	**1**	**1**	
y :	...	0	0	0	0	**1**	**0**	**0**	**1**	*i*'th bit of *y*
	...	0	0	0	**1**	**1**	**0**	**1**	**1**	1
	...	0	0	**1**	**1**	**0**	**1**	**1**	0	0
	...	0	**1**	**1**	**0**	**1**	**1**	0	0	0
...	...	**1**	**1**	**0**	**1**	**1**	0	0	0	1
x*y :	...	1	1	1	1	0	0	1	1	sum

Dijeljenje (DZ)

```
// Returns the integer part of x / y,
// where x ≥ 0 and y > 0
divide (x, y):
    if  (y > x) return 0
    q = divide (x, 2 * y)
    if ((x – 2 * q * y) < y)
        return 2 * q
    else
        return 2 * q + 1
```

```
divide(158, 3):
  q = divide(158, 6):
    q = divide(158, 12):
      q = divide(158, 24):
        q = divide(158, 48):
          q = divide(158, 96):
            q = divide(158, 192):
```

Korjenovanje

- funkcija korijen je monotono rastuća funkcija sa inverzom koji se može efikasno izračunati

- **ideja**: binarno pretraživanje!

// Compute the integer part of y = sqrt(y)

// Strategy: find an integer y such that $y^2 < x < (y+1)^2$

// by performing a binary search in the range $0, \dots, 2^{\frac{n}{2}} - 1$

sqrt (x) :

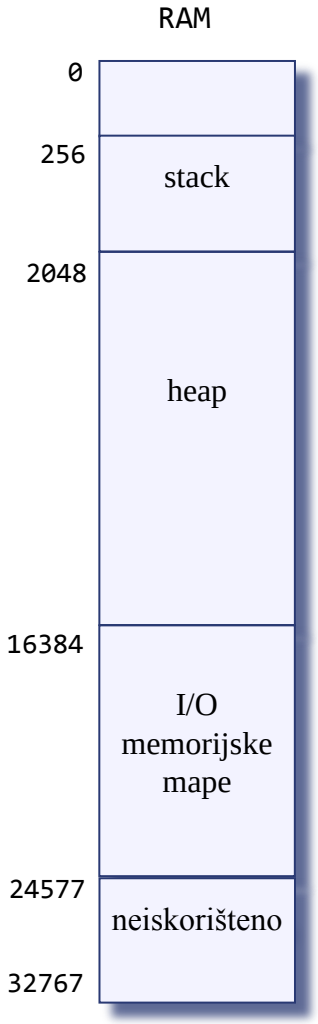
$y = 0$

for $j = \frac{n}{2} - 1, \dots, 0$ do

if $\left(y + 2^j\right)^2 < x$ then $y = y + 2^j$

return y

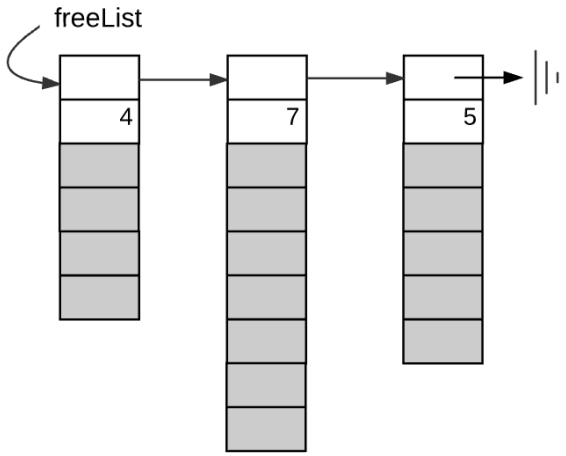
Upravljanje memorijom

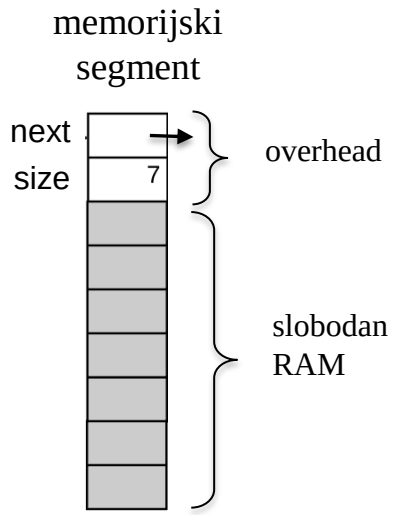



```
class Memory
{
   static array ram;
   ...

   function void init()
   {
      let ram = 0;
      ...
   }

   ...
}


// To set RAM[addr] to val:
   let ram[addr] = val;
```

Upravljanje hrpom

- koristimo povezanu listu kako bismo pratili slobodne memorijske segmente
- svaki memorijski segment sadrži pokazivač na idući segment i informaciju o svojoj duljini
- **inicijalno**: lista sadrži **cijeli** heap!
- **alokacija**: pronaći memorijski segment odgovarajuće duljine
- **dealokacija**: dodati objekt na kraj liste





Alokacija

- kako pronaći memorijski segment odgovarajuće duljine? **Heuristika**!

- **fist fit**: prvi segment koji je dovoljno dugačak
- **best fit**: najmanji mogući segment

- **postupak**: iz originalnog segmenta izrezujemo segment duljine *size + 2*