# Punë Laboratori 8

**Ushtrim 1:**

Ndërtoni një program në assembler që ekzekuton kodin e dhënë më poshtë në gjuhën C. (Vektori të jetë variabël global).

```
for (k = 0; k < count; k++) {
       for (index = 0; index < length; index ++) {
if(index < 15)
          array[index] = 0;
else
          array[index] = array[index] + 1;
      }
    }
```

Programi fillon si më poshtë (madhësia e matricës dhe vlera e count do të varet nga skenarët që paraqiten në vijim).

```
.data
array: .word 0 : 256 #matrica array ka vend per 256 fjale
length: .word 256 count: .word 4 .text .globl main
```

Si fillim vendosni me anë të një cikli gjithë elementët e matricës në vlerën 10. Më pas shkruani kodin për ciklin e dhënë më sipër.

Përdorni simulatorin Mars. Shpjegimin për përdorimin e tij e gjeni tek dokumenti ***Mars Tutorial.pdf*** (tek faqja e laboratoreve)

Ushtrimi 1.1

```
.data
array: .word 0: 256      #matrica array ka vend per 256 fjale
length: .word 256
count: .word 4

.text
.globl main

main:
          # Inicializo të gjithë elementët e array në vlerën 10
          la $t0, array        # Ngarko adresën e fillimit të array në $t0
          li $t1, 10           # Vendos vlerën për inicializimin e vektorit (10) në
      $t1
          li $t2, 256          # Ngarko gjatësinë e vektorit (256 elementë) në $t2

loop_init:
          sw $t1, 0($t0)       # Ruaj vlerën 10 në pozicionin aktual të treguar nga
      $t0
          addi $t0, $t0, 4     # Kalo te pozicioni tjetër në vektor (rrit adresën
      me 4 byte)
          addi $t2, $t2, -1    # Zvogëlo numrin e iteracioneve
          bnez $t2, loop_init  # Nëse $t2 nuk është zero, vazhdo ciklin

          # Fundi i programit
          li $v0, 10           # Ngarko kodin e daljes për syscall për të
      përfunduar programin
          syscall              # Kryej thirrjen e sistemit për të dalë
```

Ushtrimi 1.2

```
.data
vektor: .word 0: 256      # Matrica vektor ka vend për 256 fjale
length: .word 256
count: .word 4

.text
.globl main

main:
    # Inicializo të gjithë elementët e vektorit në vlerën 10
    la $t0, vektor        # Ngarko adresën e fillimit të vektorit në $t0
    li $t1, 10            # Vendos vlerën për inicializimin e vektorit (10) në $t1
    li $t2, 256           # Ngarko gjatësinë e vektorit (256 elementë) në $t2

loop_init:
    sw $t1, 0($t0)        # Ruaj vlerën 10 në pozicionin aktual të treguar nga $t0
```

```
    addi $t0, $t0, 4      # Kalo te pozicioni tjetër në vektor (rrit adresën me 4
byte)
    addi $t2, $t2, -1     # Zvogëlo numrin e iteracioneve
    bnez $t2, loop_init   # Nëse $t2 nuk është zero, vazhdo ciklin

    # Marrim vlerat e length dhe count
    lw $t3, length        # $t3 = length
    lw $t4, count         # $t4 = count
    # Cikli i jashtëm për k
    li $t5, 0             # $t5 = k

outer_loop:
    beq $t5, $t4, end_program  # Nëse k == count, përfundo programin
    # Cikli i brendshëm për index
    li $t6, 0             # $t6 = index
    la $t0, vektor        # $t0 = adresa bazë e vektorit

inner_loop:
    beq $t6, $t3, outer_loop_end  # Nëse index == length, përfundo ciklin e
brendshëm
    # Kontrolli nëse index < 15
    li $t7, 15            # $t7 = 15
    bge $t6, $t7, else_part  # Nëse index >= 15, shko te else_part
    # Pjesa if: array[index] = 0
    sw $zero, 0($t0)      # Vendos 0 në array[index]
    j skip_else

else_part:
    # Pjesa else: array[index] = array[index] + 1
    lw $t8, 0($t0)        # Lexo array[index] në $t8
    addi $t8, $t8, 1      # $t8 = array[index] + 1
    sw $t8, 0($t0)        # Ruaj vlerën e re në vektor[index]

skip_else:
    addi $t6, $t6, 1      # index++
    addi $t0, $t0, 4      # Lëviz te elementi tjetër në vektor
    j inner_loop

outer_loop_end:
    addi $t5, $t5, 1      # k++
    j outer_loop

end_program:
    # Përfundimi i programit
    li $v0, 10           # Ngarko kodin e daljes për syscall për të përfunduar
programin
    syscall               # Kryej thirrjen e sistemit për të dalë
```

**Skenari 1:**

**Cache Parameters:**

- **Placement Policy:** Direct Mapping
- **Block Replacement Policy:** LRU
- **Set size (blocks):** 1 (Mund të jetë më e madhe se 1? Pse?)
- **Number of blocks:** 4
- **Cache block size (words):** 2

**Program Parameters:**

- **Array Size:** 128 words
- **Count:** 4

Pyetje:

1. Shpjegoni hit rate që përftohet.
2. Po nëse rritet count, çfarë do të ndodhte me hit rate? Pse?
3. Si mund të modifikohen parametrat e programit që të maksimizohet hit rate? Shpjegoni përgjigjen tuaj.
4. Si mund të modifikohen parametrat e cache-së që të maksimizohet hit rate? Shpjegoni përgjigjen tuaj.

Pyetja 1

Direct Mapped Cache ka çdo bllok të cache të caktuar në një vend specifik në kujtesë. Në këtë rast, ne kemi një cache me 4 blloqe dhe madhësia e bllokut është 2 fjalë.

Array ka 128 fjalë që do të thotë 64 blloqe (128 / 2 = 64). Meqenëse cache ka vetëm 4 blloqe, çdo bllok në kujtesë do të caktohet në një nga këto 4 blloqe të cache, duke përdorur index % 4.

Nëse count = 4, Programi do të kalojë aksesojë elementët e array 4 herë. Meqenëse array ka 64 blloqe dhe cache ka vetëm 4 blloqe, çdo bllok i ri që kërkohet ne cache do të rezultojë në një miss pas herës së parë që ky bllok ka qenë në cache.Meqë aksesimi është sekuencial ne kod dhe në cache blloqet që kanë ardhur do të zëvendësohen dhe do të vijnë në cache përsëri në iteracionin pasardhës.

Pra, meqë politika e vendosjes është Direct Mapping dhe sepse aksesimi është sekuencial do të kemi miss për çdo bllok që kërkohet në cache.Pra fjala e parë në bllok është miss, kurse e dyta është hit.

Pra  Total Access: 128 * 4 = 512 access  dhe Hit Rate = Miss Rate = (64×4)/(128×4) = 0.5 = 50 %

Pyetja 2

Nëse count rritet, meqë politika është Direct Mapping , nuk do të kemi ndryshim të hit rate apo miss rate. Do të rriten herët që ndodh aksesimi i blloqeve, por nuk dryshon fakti që word-i i parë do të jetë gjithmonë miss dhe i dyti hit.

Pyetja 3

- Madhësia e Array: Duke zvogëluar madhësinë e array, do të ketë më pak blloqe për të akesesuar, dhe mundësia për hit në cache do të rritet.Psh nëse marrim Array size = 8. Herën e parë do të kemi miss dhe hit pra, 0.5 hit dhe miss.Herët e tjera blloqet kanë ardhur, pra do të kemi vetëm hit. Që të kemi përmirësim duhet që të paktën  Array Size të jetë më e vogël se 16(8 blloqe).

Pyetja 4

- Rritja e Numrit të Blloqeve: Pra rritja e madhësisë së cache.Duke rritur numrin e blloqeve në cache, më shumë blloqe të array mund të mbahen në cache njëkohësisht, duke reduktuar miss rate.
- Madhësia e Bllokut: Rritja e madhësisë së bllokut nga 2 fjalë në më shumë do të thotë që çdo bllok i cache do të përmbajë më shumë fjalë të array. Kjo do të reduktojë numrin e aksesimeve në kujtesë kryesore dhe do të rrisë hit rate.