Άσκηση 1η – Μέρος Α

Ζητούμενο: Δίνεται το παρακάτω πρόγραμμα σε C καθώς και μια μετάφραση του σε assembly MIPS. Δίνεται ότι ο καταχωρητής $s0 δείχνει στο πρώτο στοιχείο του πίνακα a και ότι οι καταχωρητές $s1 και $s2 έχουν τιμή 2 και 192 αντίστοιχα. Συμπληρώστε τα κενά. Σας υπενθυμίζουμε ότι ο καταχωρητής $0 (ή $zero) είναι πάντα μηδέν.

```
int a[192], i, sum, *p;

sum = 0;                    // $t1
p = &a[1];                  // $t2
for (i=1; i < 192; i++) {   // $t0
    if (i % 2 == 0)
        *p = sum + 13;
    else
        *p = sum + 17;

    sum += *p;
    p++;
}

a[0] = sum;
```

```
        addi  $t0,$zero, 1
        add   $t1, $zero, $zero
        addi  $t2, $s0, 4
LOOP:   slt   $t3, $t0, $s2
        beq   $t3,$zero, END
        div   $t0 , $s1
        mfhi  $t3
        bne   $t3,$zero, ELSE
IF:     addi  $t4, $t1, 13
        sw    $t4, 0($t2)
        jmp   NEXT_I
ELSE:   addi  $t4, $t1, 17
        sw    $t4, 0($t2)
NEXT_I: lw    $t3, 0($t2)
        add   $t1, $t1, $t3
        addi  $t2, $t2, 4
        addi  $t0, $t0, 1
        jmp   LOOP
END:    sw    $t1, 0($s0)
```

Άσκηση 1η – Μέρος Β

Ζητούμενο: Υλοποιήστε την παρακάτω ρουτίνα σε assembly του MIPS. Δίνονται τα μεγέθη των τύπων char 8 bits, short int 16 bits και int 32 bits.

```
struct S {
  char A[4];
  short int B[4];
  int C[4];
};

void foo(struct S *s) {
  char a;
  short int b;
  int c, i;

  a = 0; b = 0; c = 0;

  for (i=0; i < 4; i++) a += s->A[i];
  for (i=0; i < 4; i++) b += s->B[i];
  c = 0xdeadbeef;
  for (i=0; i < 4; i++) c += s->C[i];

  if (c % 50 < 20)
    c = c / 50;
}
```

```
FOO:
   add $t0, $zero, $zero
   add $t1, $zero, $zero
   add $t2, $zero, $zero

   add $t4, $zero, $zero
   addi $t5, $zero, 4
LOOP1:
   slt $t6, $t4, $t5
   beq $t6, $zero, LOOP1_END
   addi $t6, $zero, $a0
   addi $t6, $t6, $t4
   lb $t7, 0($t6)
   add $t0, $t0, $t7
   addi $t4, $t4, 1
   j LOOP1

LOOP1_END:
   addi $t4, $zero, $zero
LOOP2:
   slt $t6, $t4, $t5
   beq $t6, $zero, LOOP2_END
   sll $t6, $t4, 1
   add $t6, $t6, $a0
   addi $t6, $t6, 4
   lh $t7, 0($t6)
   add $t1, $t1, $t7
   addi $t4, $t4, 1
   j LOOP2
```

```
LOOP2_END:
   lui $t2, 0xdead
   ori $t2, 0xbeef

   addi $t4, $zero, $zero
LOOP3:
   slt $t6, $t4, $t5
   beq $t6, $zero, LOOP3_END
   sll $t6, $t4, 2
   add $t6, $t6, $a0
   addi $t6, $t6, 12
   lw $t7, 0($t6)
   add $t2, $t2, $t7
   addi $t4, $t4, 1
   j LOOP3

LOOP3_END:
   addi $t4, $zero, 50
   addi $t5, $zero, 20
   div $t2, $t4
   mfhi $t4
   slt $t6, $t4, $t5
   beq $t6, $zero, END
   mflo $t2

END:
   jr $ra
```

Άσκηση 1η – Μέρος Β

Ζητούμενο: Υλοποιήστε την παρακάτω ρουτίνα σε assembly του MIPS. Δίνονται τα μεγέθη των τύπων char 8 bits, short int 16 bits και int 32 bits.

```
struct S {
  char A[4];
  short int B[4];
  int C[4];
};

void foo(struct S *s) {
  char a;
  short int b;
  int c, i;
  a = 0; b = 0; c = 0;

  for (i=0; i < 4; i++) a += s->A[i];
  for (i=0; i < 4; i++) b += s->B[i];
  c = 0xdeadbeef;
  for (i=0; i < 4; i++) c += s->C[i];

  if (c % 50 < 20)
    c = c / 50;
}
```

$t0 $t1 $t3

```
FOO:
   add $t0, $zero, $zero
   add $t1, $zero, $zero
   add $t2, $zero, $zero

   add $t4, $zero, $zero
   addi $t5, $zero, 4
LOOP1:
   slt $t6, $t4, $t5
   beq $t6, $zero, LOOP1_END
   addi $t6, $zero, $a0
   addi $t6, $t6, $t4
   lb $t7, 0($t6)
   add $t0, $t0, $t7
   addi $t4, $t4, 1
   j LOOP1

LOOP1_END:
   addi $t4, $zero, $zero
LOOP2:
   slt $t6, $t4, $t5
   beq $t6, $zero, LOOP2_END
   sll $t6, $t4, 1
   add $t6, $t6, $a0
   addi $t6, $t6, 4
   lh $t7, 0($t6)
   add $t1, $t1, $t7
   addi $t4, $t4, 1
   j LOOP2

LOOP2_END:
   lui $t2, 0xdead
   ori $t2, 0xbeef

   addi $t4, $zero, $zero
LOOP3:
   slt $t6, $t4, $t5
   beq $t6, $zero, LOOP3_END
   sll $t6, $t4, 2
   add $t6, $t6, $a0
   addi $t6, $t6, 12
   lw $t7, 0($t6)
   add $t2, $t2, $t7
   addi $t4, $t4, 1
   j LOOP3

LOOP3_END:
   addi $t4, $zero, 50
   addi $t5, $zero, 20
   div $t2, $t4
   mfhi $t4
   slt $t6, $t4, $t5
   beq $t6, $zero, END
   mflo $t2

END:
   jr $ra
```

Άσκηση 1η – Μέρος Β

Ζητούμενο: Υλοποιήστε την παρακάτω ρουτίνα σε assembly του MIPS. Δίνονται τα μεγέθη των τύπων char 8 bits, short int 16 bits και int 32 bits.

```
struct S {
  char A[4];
  short int B[4];
  int C[4];
};

void foo(struct S *s) {
  char a;
  short int b;
  int c, i;
  a = 0; b = 0; c = 0;

  for (i=0; i < 4; i++) a += s->A[i];
  for (i=0; i < 4; i++) b += s->B[i];
  c = 0xdeadbeef;
  for (i=0; i < 4; i++) c += s->C[i];

  if (c % 50 < 20)
    c = c / 50;
}
```

$t0 **$t1** **$t3**

```
FOO:
    add $t0, $zero, $zero
    add $t1, $zero, $zero
    add $t2, $zero, $zero

    add $t4, $zero, $zero
    addi $t5, $zero, 4
LOOP1:
    slt $t6, $t4, $t5
    beq $t6, $zero, LOOP1_END
    addi $t6, $zero, $a0
    addi $t6, $t6, $t4
    lb $t7, 0($t6)
    add $t0, $t0, $t7
    addi $t4, $t4, 1
    j LOOP1

LOOP1_END:
    addi $t4, $zero, $zero
LOOP2:
    slt $t6, $t4, $t5
    beq $t6, $zero, LOOP2_END
    sll $t6, $t4, 1
    add $t6, $t6, $a0
    addi $t6, $t6, 4
    lh $t7, 0($t6)
    add $t1, $t1, $t7
    addi $t4, $t4, 1
    j LOOP2

LOOP2_END:
    lui $t2, 0xdead
    ori $t2, 0xbeef

    addi $t4, $zero, $zero
LOOP3:
    slt $t6, $t4, $t5
    beq $t6, $zero, LOOP3_END
    sll $t6, $t4, 2
    add $t6, $t6, $a0
    addi $t6, $t6, 12
    lw $t7, 0($t6)
    add $t2, $t2, $t7
    addi $t4, $t4, 1
    j LOOP3

LOOP3_END:
    addi $t4, $zero, 50
    addi $t5, $zero, 20
    div $t2, $t4
    mfhi $t4
    slt $t6, $t4, $t5
    beq $t6, $zero, END
    mflo $t2

END:
    jr $ra
```

Άσκηση 1η – Μέρος Β

Ζητούμενο: Υλοποιήστε την παρακάτω ρουτίνα σε assembly του MIPS. Δίνονται τα μεγέθη των τύπων char 8 bits, short int 16 bits και int 32 bits.

```
struct S {
  char A[4];
  short int B[4];
  int C[4];
};

void foo(struct S *s) {
  char a;
  short int b;
  int c, i;
  a = 0; b = 0; c = 0;

  for (i=0; i < 4; i++) a += s->A[i];
  for (i=0; i < 4; i++) b += s->B[i];
  c = 0xdeadbeef;
  for (i=0; i < 4; i++) c += s->C[i];

  if (c % 50 < 20)
    c = c / 50;
}
```

$t0 $t1 $t3

```
FOO:
    add $t0, $zero, $zero
    add $t1, $zero, $zero
    add $t2, $zero, $zero

    add $t4, $zero, $zero
    addi $t5, $zero, 4
LOOP1:
    slt $t6, $t4, $t5
    beq $t6, $zero, LOOP1_END
    addi $t6, $zero, $a0
    addi $t6, $t6, $t4
    lb $t7, 0($t6)
    add $t0, $t0, $t7
    addi $t4, $t4, 1
    j LOOP1

LOOP1_END:
    addi $t4, $zero, $zero
LOOP2:
    slt $t6, $t4, $t5
    beq $t6, $zero, LOOP2_END
    sll $t6, $t4, 1
    add $t6, $t6, $a0
    addi $t6, $t6, 4
    lh $t7, 0($t6)
    add $t1, $t1, $t7
    addi $t4, $t4, 1
    j LOOP2

LOOP2_END:
    lui $t2, 0xdead
    ori $t2, 0xbeef

    addi $t4, $zero, $zero
LOOP3:
    slt $t6, $t4, $t5
    beq $t6, $zero, LOOP3_END
    sll $t6, $t4, 2
    add $t6, $t6, $a0
    addi $t6, $t6, 12
    lw $t7, 0($t6)
    add $t2, $t2, $t7
    addi $t4, $t4, 1
    j LOOP3

LOOP3_END:
    addi $t4, $zero, 50
    addi $t5, $zero, 20
    div $t2, $t4
    mfhi $t4
    slt $t6, $t4, $t5
    beq $t6, $zero, END
    mflo $t2

END:
    jr $ra
```

Άσκηση 1η – Μέρος Β

Ζητούμενο: Υλοποιήστε την παρακάτω ρουτίνα σε assembly του MIPS. Δίνονται τα μεγέθη των τύπων char 8 bits, short int 16 bits και int 32 bits.

```
struct S {
  char A[4];
  short int B[4];
  int C[4];
};

void foo(struct S *s) {
  char a;
  short int b;
  int c, i;
```

$t0 **$t1** **$t3**

```
  a = 0; b = 0; c = 0;

  for (i=0; i < 4; i++) a += s->A[i];
  for (i=0; i < 4; i++) b += s->B[i];
  c = 0xdeadbeef;
  for (i=0; i < 4; i++) c += s->C[i];

  if (c % 50 < 20)
    c = c / 50;
}
```

```
FOO:
    add $t0, $zero, $zero
    add $t1, $zero, $zero
    add $t2, $zero, $zero

    add $t4, $zero, $zero
    addi $t5, $zero, 4
LOOP1:
    slt $t6, $t4, $t5
    beq $t6, $zero, LOOP1_END
    addi $t6, $zero, $a0
    addi $t6, $t6, $t4
    lb $t7, 0($t6)
    add $t0, $t0, $t7
    addi $t4, $t4, 1
    j LOOP1

LOOP1_END:
    addi $t4, $zero, $zero
LOOP2:
    slt $t6, $t4, $t5
    beq $t6, $zero, LOOP2_END
    sll $t6, $t4, 1
    add $t6, $t6, $a0
    addi $t6, $t6, 4
    lh $t7, 0($t6)
    add $t1, $t1, $t7
    addi $t4, $t4, 1
    j LOOP2

LOOP2_END:
    lui $t2, 0xdead
    ori $t2, 0xbeef

    addi $t4, $zero, $zero
LOOP3:
    slt $t6, $t4, $t5
    beq $t6, $zero, LOOP3_END
    sll $t6, $t4, 2
    add $t6, $t6, $a0
    addi $t6, $t6, 12
    lw $t7, 0($t6)
    add $t2, $t2, $t7
    addi $t4, $t4, 1
    j LOOP3

LOOP3_END:
    addi $t4, $zero, 50
    addi $t5, $zero, 20
    div $t2, $t4
    mfhi $t4
    slt $t6, $t4, $t5
    beq $t6, $zero, END
    mflo $t2

END:
    jr $ra
```

Άσκηση 1η – Μέρος Β

Ζητούμενο: Υλοποιήστε την παρακάτω ρουτίνα σε assembly του MIPS. Δίνονται τα μεγέθη των τύπων char 8 bits, short int 16 bits και int 32 bits.

```
struct S {
  char A[4];
  short int B[4];
  int C[4];
};

void foo(struct S *s) {
  char a;
  short int b;
  int c, i;
  a = 0; b = 0; c = 0;

  for (i=0; i < 4; i++) a += s->A[i];
  for (i=0; i < 4; i++) b += s->B[i];
  c = 0xdeadbeef;
  for (i=0; i < 4; i++) c += s->C[i];

  if (c % 50 < 20)
    c = c / 50;
}
```

$t0 $t1 $t3

```
FOO:
    add $t0, $zero, $zero
    add $t1, $zero, $zero
    add $t2, $zero, $zero

    add $t4, $zero, $zero
    addi $t5, $zero, 4
LOOP1:
    slt $t6, $t4, $t5
    beq $t6, $zero, LOOP1_END
    addi $t6, $zero, $a0
    addi $t6, $t6, $t4
    lb $t7, 0($t6)
    add $t0, $t0, $t7
    addi $t4, $t4, 1
    j LOOP1

LOOP1_END:
    addi $t4, $zero, $zero
LOOP2:
    slt $t6, $t4, $t5
    beq $t6, $zero, LOOP2_END
    sll $t6, $t4, 1
    add $t6, $t6, $a0
    addi $t6, $t6, 4
    lh $t7, 0($t6)
    add $t1, $t1, $t7
    addi $t4, $t4, 1
    j LOOP2

LOOP2_END:
    lui $t2, 0xdead
    ori $t2, 0xbeef

    addi $t4, $zero, $zero
LOOP3:
    slt $t6, $t4, $t5
    beq $t6, $zero, LOOP3_END
    sll $t6, $t4, 2
    add $t6, $t6, $a0
    addi $t6, $t6, 12
    lw $t7, 0($t6)
    add $t2, $t2, $t7
    addi $t4, $t4, 1
    j LOOP3

LOOP3_END:
    addi $t4, $zero, 50
    addi $t5, $zero, 20
    div $t2, $t4
    mfhi $t4
    slt $t6, $t4, $t5
    beq $t6, $zero, END
    mflo $t2

END:
    jr $ra
```

Άσκηση 1η – Μέρος Β

Ζητούμενο: Υλοποιήστε την παρακάτω ρουτίνα σε assembly του MIPS. Δίνονται τα μεγέθη των τύπων char 8 bits, short int 16 bits και int 32 bits.

```
struct S {
  char A[4];
  short int B[4];
  int C[4];
};

void foo(struct S *s) {
  char a;
  short int b;
  int c, i;
```

$t0 **$t1** **$t3**

```
  a = 0; b = 0; c = 0;

  for (i=0; i < 4; i++) a += s->A[i];
  for (i=0; i < 4; i++) b += s->B[i];
  c = 0xdeadbeef;
  for (i=0; i < 4; i++) c += s->C[i];

  if (c % 50 < 20)
    c = c / 50;
}
```

```
FOO:
    add $t0, $zero, $zero
    add $t1, $zero, $zero
    add $t2, $zero, $zero

    add $t4, $zero, $zero
    addi $t5, $zero, 4
LOOP1:
    slt $t6, $t4, $t5
    beq $t6, $zero, LOOP1_END
    addi $t6, $zero, $a0
    addi $t6, $t6, $t4
    lb $t7, 0($t6)
    add $t0, $t0, $t7
    addi $t4, $t4, 1
    j LOOP1

LOOP1_END:
    addi $t4, $zero, $zero
LOOP2:
    slt $t6, $t4, $t5
    beq $t6, $zero, LOOP2_END
    sll $t6, $t4, 1
    add $t6, $t6, $a0
    addi $t6, $t6, 4
    lh $t7, 0($t6)
    add $t1, $t1, $t7
    addi $t4, $t4, 1
    j LOOP2

LOOP2_END:
    lui $t2, 0xdead
    ori $t2, 0xbeef

    addi $t4, $zero, $zero
LOOP3:
    slt $t6, $t4, $t5
    beq $t6, $zero, LOOP3_END
    sll $t6, $t4, 2
    add $t6, $t6, $a0
    addi $t6, $t6, 12
    lw $t7, 0($t6)
    add $t2, $t2, $t7
    addi $t4, $t4, 1
    j LOOP3

LOOP3_END:
    addi $t4, $zero, 50
    addi $t5, $zero, 20
    div $t2, $t4
    mfhi $t4
    slt $t6, $t4, $t5
    beq $t6, $zero, END
    mflo $t2

END:
    jr $ra
```

Άσκηση 1η – Μέρος Β

Ζητούμενο: Υλοποιήστε την παρακάτω ρουτίνα σε assembly του MIPS. Δίνονται τα μεγέθη των τύπων char 8 bits, short int 16 bits και int 32 bits.

```
struct S {
  char A[4];
  short int B[4];
  int C[4];
};

void foo(struct S *s) {
  char a;
  short int b;
  int c, i;
  a = 0; b = 0; c = 0;

  for (i=0; i < 4; i++) a += s->A[i];
  for (i=0; i < 4; i++) b += s->B[i];
  c = 0xdeadbeef;
  for (i=0; i < 4; i++) c += s->C[i];

  if (c % 50 < 20)
    c = c / 50;
}
```

$t0 $t1 $t3

```
FOO:
   add $t0, $zero, $zero
   add $t1, $zero, $zero
   add $t2, $zero, $zero

   add $t4, $zero, $zero
   addi $t5, $zero, 4
LOOP1:
   slt $t6, $t4, $t5
   beq $t6, $zero, LOOP1_END
   addi $t6, $zero, $a0
   addi $t6, $t6, $t4
   lb $t7, 0($t6)
   add $t0, $t0, $t7
   addi $t4, $t4, 1
   j LOOP1

LOOP1_END:
   addi $t4, $zero, $zero
LOOP2:
   slt $t6, $t4, $t5
   beq $t6, $zero, LOOP2_END
   sll $t6, $t4, 1
   add $t6, $t6, $a0
   addi $t6, $t6, 4
   lh $t7, 0($t6)
   add $t1, $t1, $t7
   addi $t4, $t4, 1
   j LOOP2

LOOP2_END:
   lui $t2, 0xdead
   ori $t2, 0xbeef

   addi $t4, $zero, $zero
LOOP3:
   slt $t6, $t4, $t5
   beq $t6, $zero, LOOP3_END
   sll $t6, $t4, 2
   add $t6, $t6, $a0
   addi $t6, $t6, 12
   lw $t7, 0($t6)
   add $t2, $t2, $t7
   addi $t4, $t4, 1
   j LOOP3

LOOP3_END:
   addi $t4, $zero, 50
   addi $t5, $zero, 20
   div $t2, $t4
   mfhi $t4
   slt $t6, $t4, $t5
   beq $t6, $zero, END
   mflo $t2

END:
   jr $ra
```

Άσκηση 1η – Μέρος Γ

Ζητούμενο: Οι παρακάτω ρουτίνες σε C υλοποιούν τους αλγορίθμους γραμμικής και δυαδικής αναζήτησης σε έναν πίνακα ακεραίων με N στοιχεία. Οι δύο ρουτίνες επιστρέφουν τη θέση μέσα στον πίνακα που βρίσκεται ο ακέραιος που αναζητούμε ή -1 σε περίπτωση που ο ακέραιος δεν βρεθεί. Υλοποιήστε τις ρουτίνες σε assembly του MIPS.

```
int linear_search(int *A, int N, int key)
{
  int i;
  for (i=0; i < N; i++)
    if (A[i] == key)
      return i;
  return -1;
}

int binary_search_rec(int *A, int l, int r, int key) {
  int mid;

  if (r < l)
    return -1;

  mid = l + (r - l) / 2;
  if (A[mid] == key)
    return mid;
  else if (A[mid] > key)
    return binary_search_rec(A, l, mid-1, key);
  else
    return binary_search_rec(A, mid+1, r, key);
}

int binary_search(int *A, int N, int key) {
  return binary_search_rec(A, 0, N-1, key);
}
```

Άσκηση 1η – Μέρος Γ

```
int linear_search(int *A, int N, int key)
{
  int i;
  for (i=0; i < N; i++)
    if (A[i] == key)
      return i;
  return -1;
}
```

```
LINEAR_SEARCH:
   add $t0, $zero, $zero

   LOOP:
      slt $t1, $t0, $a1
      beq $t1, $zero, END

      sll $t2, $t0, 2
      add $t2, $t2, $a0
      lw $t3, 0($t2)
      slt $t1, $t3, $a2
      bne $t1, $zero, NEXT_I
      add $v0, $t0, $zero
      jr $ra

   NEXT_I:
      addi $t0, $t0, 1
      j LOOP

   END:
      addi $v0, $zero, -1
      jr $ra
```

Άσκηση 1η – Μέρος Γ

```
int binary_search_rec(int *A, int l, int r, int key) {
  int mid;

  if (r < l)
    return -1;

  mid = l + (r - l) / 2;
  if (A[mid] == key)
    return mid;
  else if (A[mid] > key)
    return binary_search_rec(A, l, mid-1, key);
  else
    return binary_search_rec(A, mid+1, r, key);
}

int binary_search(int *A, int N, int key) {
  return binary_search_rec(A, 0, N-1, key);
}
```

```
BINARY_SEARCH_REC:
   slt $t0, $a2, $a1
   beq $t0, $zero, MAIN_BODY
   addi $v0, $zero, -1
   jr $ra

   MAIN_BODY:
      sub $t0, $a2, $a1
      srl $t0, $t0, 1
      add $t0, $t0, $a1

      sll $t2, $t0, 2
      add $t2, $t2, $a0
      lw $t1, 0($t2)
      bne $t1, $a3, ELSE_IF
      add $v0, $t0, $zero
      jr $ra

      ELSE_IF:
         addi $sp, $sp, -4
         sw $ra, 0($sp)
         slt $t2, $t1, $a3
         bne $t2, $zero, ELSE
         addi $a2, $t0, -1
         j RECURSIVE_CALL

      ELSE:
         addi $a1, $t0, 1

      RECURSIVE_CALL:
         jal binary_search_rec

         lw $ra, 0($sp)
         addi $sp, $sp, 4
         jr $ra
```

Άσκηση 1η – Μέρος Γ

```
int binary_search_rec(int *A, int l, int r, int key) {
  int mid;

  if (r < l)
    return -1;

  mid = l + (r - l) / 2;
  if (A[mid] == key)
    return mid;
  else if (A[mid] > key)
    return binary_search_rec(A, l, mid-1, key);
  else
    return binary_search_rec(A, mid+1, r, key);
}

int binary_search(int *A, int N, int key) {
  return binary_search_rec(A, 0, N-1, key);
}
```

```
BINARY_SEARCH_REC:
  slt $t0, $a2, $a1
  beq $t0, $zero, MAIN_BODY
  addi $v0, $zero, -1
  jr $ra

MAIN_BODY:
    sub $t0, $a2, $a1
    srl $t0, $t0, 1
    add $t0, $t0, $a1

    sll $t2, $t0, 2
    add $t2, $t2, $a0
    lw $t1, 0($t2)
    bne $t1, $a3, ELSE_IF
    add $v0, $t0, $zero
    jr $ra

    ELSE_IF:
      addi $sp, $sp, -4
      sw $ra, 0($sp)
      slt $t2, $t1, $a3
      bne $t2, $zero, ELSE
      addi $a2, $t0, -1
      j RECURSIVE_CALL

    ELSE:
      addi $a1, $t0, 1

    RECURSIVE_CALL:
      jal binary_search_rec

      lw $ra, 0($sp)
      addi $sp, $sp, 4
      jr $ra
```

Άσκηση 1η – Μέρος Γ

```
int binary_search_rec(int *A, int l, int r, int key) {
  int mid;

  if (r < l)
    return -1;

  mid = l + (r - l) / 2;
  if (A[mid] == key)
    return mid;
  else if (A[mid] > key)
    return binary_search_rec(A, l, mid-1, key);
  else
    return binary_search_rec(A, mid+1, r, key);
}

int binary_search(int *A, int N, int key) {
  return binary_search_rec(A, 0, N-1, key);
}
```

```
BINARY_SEARCH_REC:
   slt $t0, $a2, $a1
   beq $t0, $zero, MAIN_BODY
   addi $v0, $zero, -1
   jr $ra

MAIN_BODY:
   sub $t0, $a2, $a1
   srl $t0, $t0, 1
   add $t0, $t0, $a1

   sll $t2, $t0, 2
   add $t2, $t2, $a0
   lw $t1, 0($t2)
   bne $t1, $a3, ELSE_IF
   add $v0, $t0, $zero
   jr $ra

   ELSE_IF:
      addi $sp, $sp, -4
      sw $ra, 0($sp)
      slt $t2, $t1, $a3
      bne $t2, $zero, ELSE
      addi $a2, $t0, -1
      j RECURSIVE_CALL

   ELSE:
      addi $a1, $t0, 1

   RECURSIVE_CALL:
      jal binary_search_rec

      lw $ra, 0($sp)
      addi $sp, $sp, 4
      jr $ra
```

Άσκηση 1η – Μέρος Γ

```
int binary_search_rec(int *A, int l, int r, int key) {
  int mid;

  if (r < l)
    return -1;

  mid = l + (r - l) / 2;
  if (A[mid] == key)
    return mid;
  else if (A[mid] > key)
    return binary_search_rec(A, l, mid-1, key);
  else
    return binary_search_rec(A, mid+1, r, key);
}

int binary_search(int *A, int N, int key) {
  return binary_search_rec(A, 0, N-1, key);
}
```

```
BINARY_SEARCH_REC:
   slt $t0, $a2, $a1
   beq $t0, $zero, MAIN_BODY
   addi $v0, $zero, -1
   jr $ra

   MAIN_BODY:
      sub $t0, $a2, $a1
      srl $t0, $t0, 1
      add $t0, $t0, $a1

      sll $t2, $t0, 2
      add $t2, $t2, $a0
      lw $t1, 0($t2)
      bne $t1, $a3, ELSE_IF
      add $v0, $t0, $zero
      jr $ra

      ELSE_IF:
         addi $sp, $sp, -4
         sw $ra, 0($sp)
         slt $t2, $t1, $a3
         bne $t2, $zero, ELSE
         addi $a2, $t0, -1
         j RECURSIVE_CALL

      ELSE:
         addi $a1, $t0, 1

      RECURSIVE_CALL:
         jal binary_search_rec

         lw $ra, 0($sp)
         addi $sp, $sp, 4
         jr $ra
```

Άσκηση 1η – Μέρος Γ

```
int binary_search_rec(int *A, int l, int r, int key) {
  int mid;

  if (r < l)
    return -1;

  mid = l + (r - l) / 2;
  if (A[mid] == key)
    return mid;
  else if (A[mid] > key)
    return binary_search_rec(A, l, mid-1, key);
  else
    return binary_search_rec(A, mid+1, r, key);
}

int binary_search(int *A, int N, int key) {
  return binary_search_rec(A, 0, N-1, key);
}
```

```
BINARY_SEARCH_REC:
   slt $t0, $a2, $a1
   beq $t0, $zero, MAIN_BODY
   addi $v0, $zero, -1
   jr $ra

   MAIN_BODY:
      sub $t0, $a2, $a1
      srl $t0, $t0, 1
      add $t0, $t0, $a1

      sll $t2, $t0, 2
      add $t2, $t2, $a0
      lw $t1, 0($t2)
      bne $t1, $a3, ELSE_IF
      add $v0, $t0, $zero
      jr $ra

      ELSE_IF:
         addi $sp, $sp, -4
         sw $ra, 0($sp)
         slt $t2, $t1, $a3
         bne $t2, $zero, ELSE
         addi $a2, $t0, -1
         j RECURSIVE_CALL

      ELSE:
         addi $a1, $t0, 1

      RECURSIVE_CALL:
         jal binary_search_rec

         lw $ra, 0($sp)
         addi $sp, $sp, 4
         jr $ra
```

Άσκηση 1η – Μέρος Γ

```
int binary_search_rec(int *A, int l, int r, int key) {
  int mid;

  if (r < l)
    return -1;

  mid = l + (r - l) / 2;
  if (A[mid] == key)
    return mid;
  else if (A[mid] > key)
    return binary_search_rec(A, l, mid-1, key);
  else
    return binary_search_rec(A, mid+1, r, key);
}

int binary_search(int *A, int N, int key) {
  return binary_search_rec(A, 0, N-1, key);
}
```

```
BINARY_SEARCH_REC:
   slt $t0, $a2, $a1
   beq $t0, $zero, MAIN_BODY
   addi $v0, $zero, -1
   jr $ra

   MAIN_BODY:
      sub $t0, $a2, $a1
      srl $t0, $t0, 1
      add $t0, $t0, $a1

      sll $t2, $t0, 2
      add $t2, $t2, $a0
      lw $t1, 0($t2)
      bne $t1, $a3, ELSE_IF
      add $v0, $t0, $zero
      jr $ra

      ELSE_IF:
         addi $sp, $sp, -4
         sw $ra, 0($sp)
         slt $t2, $t1, $a3
         bne $t2, $zero, ELSE
         addi $a2, $t0, -1
         j RECURSIVE_CALL

      ELSE:
         addi $a1, $t0, 1

      RECURSIVE_CALL:
         jal binary_search_rec

         lw $ra, 0($sp)
         addi $sp, $sp, 4
         jr $ra
```

Άσκηση 1η – Μέρος Γ

```
int binary_search_rec(int *A, int l, int r, int key) {
  int mid;

  if (r < l)
    return -1;

  mid = l + (r - l) / 2;
  if (A[mid] == key)
    return mid;
  else if (A[mid] > key)
    return binary_search_rec(A, l, mid-1, key);
  else
    return binary_search_rec(A, mid+1, r, key);
}

int binary_search(int *A, int N, int key) {
  return binary_search_rec(A, 0, N-1, key);
}
```

```
BINARY_SEARCH_REC:
   slt $t0, $a2, $a1
   beq $t0, $zero, MAIN_BODY
   addi $v0, $zero, -1
   jr $ra

   MAIN_BODY:
      sub $t0, $a2, $a1
      srl $t0, $t0, 1
      add $t0, $t0, $a1

      sll $t2, $t0, 2
      add $t2, $t2, $a0
      lw $t1, 0($t2)
      bne $t1, $a3, ELSE_IF
      add $v0, $t0, $zero
      jr $ra

      ELSE_IF:
         addi $sp, $sp, -4
         sw $ra, 0($sp)
         slt $t2, $t1, $a3
         bne $t2, $zero, ELSE
         addi $a2, $t0, -1
         j RECURSIVE_CALL

      ELSE:
         addi $a1, $t0, 1

      RECURSIVE_CALL:
         jal binary_search_rec

         lw $ra, 0($sp)
         addi $sp, $sp, 4
         jr $ra
```

Άσκηση 1η – Μέρος Γ

```
int binary_search_rec(int *A, int l, int r, int key) {
  int mid;

  if (r < l)
    return -1;

  mid = l + (r - l) / 2;
  if (A[mid] == key)
    return mid;
  else if (A[mid] > key)
    return binary_search_rec(A, l, mid-1, key);
  else
    return binary_search_rec(A, mid+1, r, key);
}

int binary_search(int *A, int N, int key) {
  return binary_search_rec(A, 0, N-1, key);
}
```

```
BINARY_SEARCH:
  addi $sp, $sp, -4
  sw $ra, 0($sp)

  add $a3, $a2, $zero
  addi $a2, $a1, -1
  add $a1, $zero, $zero
  jal binary_search_rec

  lw $ra, 0($sp)
  addi $sp, $sp, 4
  jr $ra
```

2η Άσκηση

Δεδομένα

Έχουμε ένα loop...

```
LOOP: LW $t1, 0($t0)
      LW $t2, 8($t0)
      ADDI  $t2, $t2, 4
      LW $t3, 0($t2)
      ADD   $t3, $t1, $t3
      LW $t4, 4($t3)
      SW $t3, 0($t4)
      ADDI  $t0, $t0, 4
      ADDI  $t9, $t9, -4
      BNEZ  $t9, LOOP
```

2η Άσκηση

Δεδομένα

Έχουμε ένα loop...

```
LOOP: LW $t1, 0($t0)
      LW $t2, 8($t0)
      ADDI  $t2, $t2, 4
      LW $t3, 0($t2)
      ADD   $t3, $t1, $t3
      LW $t4, 4($t3)
      SW $t3, 0($t4)
      ADDI  $t0, $t0, 4
      ADDI  $t9, $t9, -4
      BNEZ  $t9, LOOP
```

και αυτή την αρχική κατάσταση

$t9 = 512

- Δεν υπάρχει cache miss
- Cache hit σε 1cc
- branches γίνονται resolve στο EX stage

2η Άσκηση

```
LOOP: LW $t1, 0($t0)
      LW $t2, 8($t0)
      ADDI  $t2, $t2, 4
      LW $t3, 0($t2)
      ADD   $t3, $t1, $t3
      LW $t4, 4($t3)
      SW $t3, 0($t4)
      ADDI  $t0, $t0, 4
      ADDI  $t9, $t9, -4
      BNEZ  $t9, LOOP
```

`$t9 = 512`

→ `$t9 = 512, 508, …, 0`

Ο βρόχος θα εκτελεστεί 512/ 4 = 128 φορές.

2η Άσκηση – 1ο ζητούμενο

1ο Ζητούμενο : Για το 1ο LOOP (μέχρι και το lw του 2ου LOOP)

Να δείξετε τα **διάφορα στάδια του pipeline** (διάγραμμα χρονισμού) που περνάνε οι εντολές. Υποθέστε ότι η αρχιτεκτονική δε διαθέτει σχήμα προώθησης.

Κύκλος	1	2	3	4	5	6	7
Εντολή 1	IF	ID	EX	MEM	WB		
Εντολή 2		IF	ID	EX	MEM	WB	
Εντολή 3			…	…	…	…	…
…							

2η Άσκηση – 1ο ζητούμενο

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
`LW $t1,0($t0)`	F																											
`LW $t2,8($t0)`																												
`ADDI $t2,$t2,4`																												
`LW $t3,0($t2)`																												
`ADD $t3,$t1,$t3`																												
`LW $t4,4($t3)`																												
`SW $t3,0($t4)`																												
`ADDI $t0,$t0,4`																												
`ADDI $t9,$t9,-4`																												
`BNEZ $t9,LOOP`																												
`LW $t1,0($t0)`																												

2η Άσκηση – 1ο ζητούμενο

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
`LW $t1,0($t0)`	F	D																										
`LW $t2,8($t0)`		F																										
`ADDI $t2,$t2,4`																												
`LW $t3,0($t2)`																												
`ADD $t3,$t1,$t3`																												
`LW $t4,4($t3)`																												
`SW $t3,0($t4)`																												
`ADDI $t0,$t0,4`																												
`ADDI $t9,$t9,-4`																												
`BNEZ $t9,LOOP`																												
`LW $t1,0($t0)`																												

2η Άσκηση – 1ο ζητούμενο

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
`LW $t1,0($t0)`	F	D	X																									
`LW $t2,8($t0)`		F	D																									
`ADDI $t2,$t2,4`			F																									
`LW $t3,0($t2)`																												
`ADD $t3,$t1,$t3`																												
`LW $t4,4($t3)`																												
`SW $t3,0($t4)`																												
`ADDI $t0,$t0,4`																												
`ADDI $t9,$t9,-4`																												
`BNEZ $t9,LOOP`																												
`LW $t1,0($t0)`																												

2η Άσκηση – 1ο ζητούμενο

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
`LW $t1,0($t0)`	F	D	X	M																								
`LW $t2,8($t0)`		F	D	X																								
`ADDI $t2,$t2,4`			F	D																								
`LW $t3,0($t2)`				F																								
`ADD $t3,$t1,$t3`																												
`LW $t4,4($t3)`																												
`SW $t3,0($t4)`																												
`ADDI $t0,$t0,4`																												
`ADDI $t9,$t9,-4`																												
`BNEZ $t9,LOOP`																												
`LW $t1,0($t0)`																												

2η Άσκηση – 1ο ζητούμενο

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
LW $t1,0($t0)	F	D	X	M	W																							
LW $t2,8($t0)		F	D	X	M																							
ADDI $t2,$t2,4			F	D	-																							
LW $t3,0($t2)				F	-																							
ADD $t3,$t1,$t3																												
LW $t4,4($t3)																												
SW $t3,0($t4)																												
ADDI $t0,$t0,4																												
ADDI $t9,$t9,-4																												
BNEZ $t9,LOOP																												
LW $t1,0($t0)																												

2η Άσκηση – 1ο ζητούμενο

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
`LW $t1,0($t0)`	F	D	X	M	W																							
`LW $t2,8($t0)`		F	D	X	M	W																						
`ADDI $t2,$t2,4`			F	D	-	-																						
`LW $t3,0($t2)`				F	-	-																						
`ADD $t3,$t1,$t3`																												
`LW $t4,4($t3)`																												
`SW $t3,0($t4)`																												
`ADDI $t0,$t0,4`																												
`ADDI $t9,$t9,-4`																												
`BNEZ $t9,LOOP`																												
`LW $t1,0($t0)`																												

2η Άσκηση – 1ο ζητούμενο

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
`LW $t1,0($t0)`	F	D	X	M	W																							
`LW $t2,8($t0)`		F	D	X	M	W																						
`ADDI $t2,$t2,4`			F	D	-	-	X																					
`LW $t3,0($t2)`				F	-	-	D																					
`ADD $t3,$t1,$t3`							F																					
`LW $t4,4($t3)`																												
`SW $t3,0($t4)`																												
`ADDI $t0,$t0,4`																												
`ADDI $t9,$t9,-4`																												
`BNEZ $t9,LOOP`																												
`LW $t1,0($t0)`																												

2η Άσκηση – 1ο ζητούμενο

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
LW $t1,0($t0)	F	D	X	M	W																							
LW $t2,8($t0)		F	D	X	M	W																						
ADDI $t2,$t2,4			F	D	-	-	X	M																				
LW $t3,0($t2)				F	-	-	D	-																				
ADD $t3,$t1,$t3							F	-																				
LW $t4,4($t3)																												
SW $t3,0($t4)																												
ADDI $t0,$t0,4																												
ADDI $t9,$t9,-4																												
BNEZ $t9,LOOP																												
LW $t1,0($t0)																												

2η Άσκηση – 1ο ζητούμενο

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
`LW $t1,0($t0)`	F	D	X	M	W																							
`LW $t2,8($t0)`		F	D	X	M	W																						
`ADDI $t2,$t2,4`			F	D	-	-	X	M	W																			
`LW $t3,0($t2)`				F	-	-	D	-	-																			
`ADD $t3,$t1,$t3`							F	-	-																			
`LW $t4,4($t3)`																												
`SW $t3,0($t4)`																												
`ADDI $t0,$t0,4`																												
`ADDI $t9,$t9,-4`																												
`BNEZ $t9,LOOP`																												
`LW $t1,0($t0)`																												

2η Άσκηση – 1ο ζητούμενο

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
`LW $t1,0($t0)`	F	D	X	M	W																							
`LW $t2,8($t0)`		F	D	X	M	W																						
`ADDI $t2,$t2,4`			F	D	-	-	X	M	W																			
`LW $t3,0($t2)`				F	-	-	D	-	-	X																		
`ADD $t3,$t1,$t3`							F	-	-	D																		
`LW $t4,4($t3)`										F																		
`SW $t3,0($t4)`																												
`ADDI $t0,$t0,4`																												
`ADDI $t9,$t9,-4`																												
`BNEZ $t9,LOOP`																												
`LW $t1,0($t0)`																												

2η Άσκηση – 1ο ζητούμενο

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
`LW $t1,0($t0)`	F	D	X	M	W																							
`LW $t2,8($t0)`		F	D	X	M	W																						
`ADDI $t2,$t2,4`			F	D	-	-	X	M	W																			
`LW $t3,0($t2)`				F	-	-	D	-	-	X	M																	
`ADD $t3,$t1,$t3`							F	-	-	D	-																	
`LW $t4,4($t3)`										F	-																	
`SW $t3,0($t4)`																												
`ADDI $t0,$t0,4`																												
`ADDI $t9,$t9,-4`																												
`BNEZ $t9,LOOP`																												
`LW $t1,0($t0)`																												

2η Άσκηση – 1ο ζητούμενο

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
`LW $t1,0($t0)`	F	D	X	M	W																							
`LW $t2,8($t0)`		F	D	X	M	W																						
`ADDI $t2,$t2,4`			F	D	-	-	X	M	W																			
`LW $t3,0($t2)`				F	-	-	D	-	-	X	M	W																
`ADD $t3,$t1,$t3`							F	-	-	D	-	-																
`LW $t4,4($t3)`										F	-	-																
`SW $t3,0($t4)`																												
`ADDI $t0,$t0,4`																												
`ADDI $t9,$t9,-4`																												
`BNEZ $t9,LOOP`																												
`LW $t1,0($t0)`																												

2η Άσκηση – 1ο ζητούμενο

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
`LW $t1,0($t0)`	F	D	X	M	W																							
`LW $t2,8($t0)`		F	D	X	M	W																						
`ADDI $t2,$t2,4`			F	D	-	-	X	M	W																			
`LW $t3,0($t2)`				F	-	-	D	-	-	X	M	W																
`ADD $t3,$t1,$t3`							F	-	-	D	-	-	X	M	W													
`LW $t4,4($t3)`										F	-	-	D	-	-	X	M	W										
`SW $t3,0($t4)`													F	-	-	D	-	-	X	M	W							
`ADDI $t0,$t0,4`																F	-	-	D	X	M	W						
`ADDI $t9,$t9,-4`																			F	D	X	M	W					
`BNEZ $t9,LOOP`																				F	D	-	-	X	M	W		
`LW $t1,0($t0)`																									F	D	X	M

Σύνολο κύκλων: 24*127 + 26 = 3074

2η Άσκηση – 2ο ζητούμενο

2ο Ζητούμενο : Για το 1ο LOOP (μέχρι και το lw του 2ου LOOP)

Να δείξετε τα **διάφορα στάδια του pipeline** (διάγραμμα χρονισμού) που περνάνε οι εντολές. Υποθέστε τώρα ότι η αρχιτεκτονική **διαθέτει σχήμα προώθησης**.

Κύκλος	1	2	3	4	5	6	7
Εντολή 1	IF	ID	EX	MEM	WB		
Εντολή 2		IF	ID	EX	MEM	WB	
Εντολή 3			…	…	…	…	…
…							

2η Άσκηση – 2ο ζητούμενο

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
LW $t1,0($t0)	F	D	X	M	W															
LW $t2,8($t0)		F	D	X	M	W														
ADDI $t2,$t2,4			F	D	-	X	M	W												
LW $t3,0($t2)				F	-	D	X	M	W											
ADD $t3,$t1,$t3						F	D	-	X	M	W									
LW $t4,4($t3)							F	-	D	X	M	W								
SW $t3,0($t4)									F	D	-	X	M	W						
ADDI $t0,$t0,4										F	-	D	X	M	W					
ADDI $t9,$t9,-4												F	-	D	X	M	W			
BNEZ $t9,LOOP														F	D	X	M	W		
LW $t1,0($t0)																	F	D	X	M

Σύνολο κύκλων: 16*127 + 18 = 2050

2η Άσκηση – 3ο ζητούμενο

3ο Ζητούμενο : Για το 1ο LOOP (μέχρι και το lw του 2ου LOOP)

Προσπαθήστε να πετύχετε καλύτερη απόδοση τροποποιώντας τον κώδικα, χωρίς όμως να αλλάξετε τη σημασιολογία του προγράμματος.

Κύκλος	1	2	3	4	5	6	7
Εντολή 1	IF	ID	EX	MEM	WB		
Εντολή 2		IF	ID	EX	MEM	WB	
Εντολή 3			…	…	…	…	…
…							

2η Άσκηση – 3ο ζητούμενο

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
`LW $t1,0($t0)`	F	D	X	M	W															
`LW $t2,8($t0)`		F	D	X	M	W														
`ADDI $t2,$t2,4`			F	D	-	X	M	W												
`LW $t3,0($t2)`				F	-	D	X	M	W											
`ADD $t3,$t1,$t3`						F	D	-	X	M	W									
`LW $t4,4($t3)`							F	-	D	X	M	W								
`SW $t3,0($t4)`									F	D	-	X	M	W						
`ADDI $t0,$t0,4`										F	-	D	X	M	W					
`ADDI $t9,$t9,-4`												F	-	D	X	M	W			
`BNEZ $t9,LOOP`														F	D	X	M	W		
`LW $t1,0($t0)`																	F	D	X	M

Σύνολο κύκλων: 16*127 + 18 = 2050

2η Άσκηση – 3ο ζητούμενο

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
LW $t1,0($t0)	F	D	X	M	W															
LW $t2,8($t0)		F	D	X	M	W														
ADDI $t2,$t2,4			F	D	-	X	M	W												
LW $t3,0($t2)				F	-	D	X	M	W											
ADD $t3,$t1,$t3						F	D	-	X	M	W									
LW $t4,4($t3)							F	-	D	X	M	W								
SW $t3,0($t4)									F	D	-	X	M	W						
ADDI $t0,$t0,4										F	-	D	X	M	W					
ADDI $t9,$t9,-4												F	-	D	X	M	W			
BNEZ $t9,LOOP														F	D	X	M	W		
LW $t1,0($t0)																	F	D	X	M

Θέλουμε να τα αποφύγουμε!

Σύνολο κύκλων: 16*127 + 18 = 2050

2η Άσκηση – 3ο ζητούμενο

Αναδιατάσσονται

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
LW $t1,0($t0)	F	D	X	M	W															
LW $t2,8($t0)		F	D	X	M	W														
ADDI $t2,$t2,4			F	D	-	X	M	W												
LW $t3,0($t2)				F	-	D	X	M	W											
ADD $t3,$t1,$t3						F	D	-	X	M	W									
LW $t4,4($t3)							F	-	D	X	M	W								
SW $t3,0($t4)									F	D	-	X	M	W						
ADDI $t0,$t0,4										F	-	D	X	M	W					
ADDI $t9,$t9,-4												F	-	D	X	M	W			
BNEZ $t9,LOOP														F	D	X	M	W		
LW $t1,0($t0)																	F	D	X	M

Θέλουμε να τα αποφύγουμε!

Σύνολο κύκλων: 16*127 + 18 = 2050

2η Άσκηση – 3ο ζητούμενο

2η Άσκηση – 3ο ζητούμενο

2η Άσκηση – 3ο ζητούμενο

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
LW $t2, 8($t0)	F	D	X	M	W												
LW $t1, 0($t0)		F	D	X	M	W											
LW $t3, 4($t2)			F	D	X	M	W										
ADDI $t2, $t2, 4				F	D	X	M	W									
ADD $t3, $t1, $t3					F	D	X	M	W								
LW $t4, 4($t3)						F	D	X	M	W							
ADDI $t9, $t9, -4							F	D	X	M	W						
SW $t3, 0($t4)								F	D	X	M	W					
ADDI $t0, $t0, 4									F	D	X	M	W				
BNEZ $t9, LOOP										F	D	X	M	W			
LW $t2, 8($t0)													F	D	X	M	W

Σύνολο κύκλων: 12*127 + 14 = 1538

Άσκηση 3η – Μέρος Α

Ζητούμενο: Δίνεται επεξεργαστής με ένα επίπεδο κρυφής μνήμης με μέσο χρόνο πρόσβασης στη μνήμη 2.4 κύκλους ρολογιού. Πιο συγκεκριμένα, τα hits εξυπηρετούνται σε 1 κύκλο ενώ το misses εξυπηρετούνται από την κύρια μνήμη σε 80 κύκλους. Σας ζητούν να προσθέσετε ένα δεύτερο επίπεδο κρυφής μνήμης ώστε η επιτάχυνση (speedup) του μέσου χρόνου πρόσβασης να είναι ίση με 1.65. Ποιό το hit rate αυτής της L2, αν η πρόσβαση σε αυτή στοιχίζει 6 κύκλους;

$$AMAT_{\mathrm{L1}} = 1 * hit_{L1} + MR_{L1} * MP_{L1} = 2.4 \Rightarrow MR_{L1} = \frac{2.4 - 1}{80} = 0.0175$$

$$AMAT_{\mathrm{L1+L2}} = 1 * hit_{L1} + MR_{L1} * (hit_{L2} + MR_{L2} * MP_{L2})$$

$$\frac{AMAT_{L1}}{AMAT_{L1+L2}} = 1.65 \Rightarrow \frac{2.4}{1 * hit_{L1} + MR_{L1} * (hit_{L2} + MR_{L2} * MP_{L2})} = 1.65$$

$$\Rightarrow \frac{2.4}{1 + 0.0175(6 + 80 * MR_{L2})} = 1.65 \Rightarrow \boldsymbol{MR_{L2} = 0.25}$$

L2 Hit Rate = 75%

Άσκηση 3η – Μέρος Β

Δίνεται το ακόλουθο κομμάτι κώδικα:

```
int i,j;
double A[8][8], B[16];

for (i=0; i < 5; i++)
   for (j=0; j < 8; j++)
      A[i][j] = A[i+1][j] + A[i+2][j] + B[j] + B[j+8];
```

- ✓ *Κάθε στοιχείο του πίνακα είναι 8 bytes*
- ✓ *1 επίπεδο κρυφής μνήμης, 2-way associative, LRU πολιτική αντικατάστασης, write-allocate, 32B block, μεγέθους 256 bytes*
- ✓ *Αρχικά η cache είναι άδεια*
- ✓ *Όλες οι μεταβλητές αποθηκεύονται σε καταχωρητές εκτός από τα στοιχεία των πινάκων*
- ✓ *Οι πίνακες αποθηκεύονται στη μνήμη κατά γραμμές και είναι ευθυγραμμισμένοι*
- ✓ *Ο Α στη θέση μνήμης 0x00008000*
- ✓ *Η σειρά με την οποία γίνονται οι αναφορές στην μνήμη είναι Α, Β, Α*

Άσκηση 3η – Μέρος Β (i)

1ο Ζητούμενο: Βρείτε το συνολικό αριθμό hits και misses για όλη την εκτέλεση του παραπάνω κώδικα.

- 1 block = 32 bytes
- 1 στοιχείο = 8 bytes
- πίνακας αποθηκευμένος κατά γραμμές

→ σε 1 block της cache θα απεικονίζονται 4 διαδοχικά στοιχεία του πίνακα, π.χ. A[i][j],A[i][j+1], A[i][j+2], A[i][j+3]

32 bytes block size = 2^5 → 5 bits offset
256B cache / 32B block → 8 blocks
8 blocks → 4 sets = 2^2 → 2 bits index
Πίνακας Α αποθηκευμένος στη θέσή 0x8000
Πίνακας Β αποθηκευμένος μετά τον Α → 0x8000 + 0x200 (=512) → 0x8200

A[0][0] → 0x0000 8000 = 0000 0000 0000 0000 0001 0000 0000 0000
B[0] → 0x0008 0200 = 0000 0000 0000 0000 0001 0010 0000 0000

A[0][0] και B[0] γίνονται mapped στο ίδιο block

Άσκηση 3η – Μέρος Β (i)

1ο Ζητούμενο: Βρείτε το συνολικό αριθμό hits και misses για όλη την εκτέλεση του παραπάνω κώδικα.

- 1 block = 32 bytes
- 1 στοιχείο = 8 bytes
- πίνακας αποθηκευμένος κατά γραμμές

→ σε 1 block της cache θα απεικονίζονται 4 διαδοχικά στοιχεία του πίνακα, π.χ. A[i][j],A[i][j+1], A[i][j+2], A[i][j+3]

32 bytes block size = 2^5 → 5 bits offset
256B cache / 32B block → 8 blocks
8 blocks → 4 sets = 2^2 → 2 bits index
Πίνακας Α αποθηκευμένος στη θέσή 0x8000
Πίνακας Β αποθηκευμένος μετά τον Α → 0x8000 + 0x200 (=512) → 0x8200

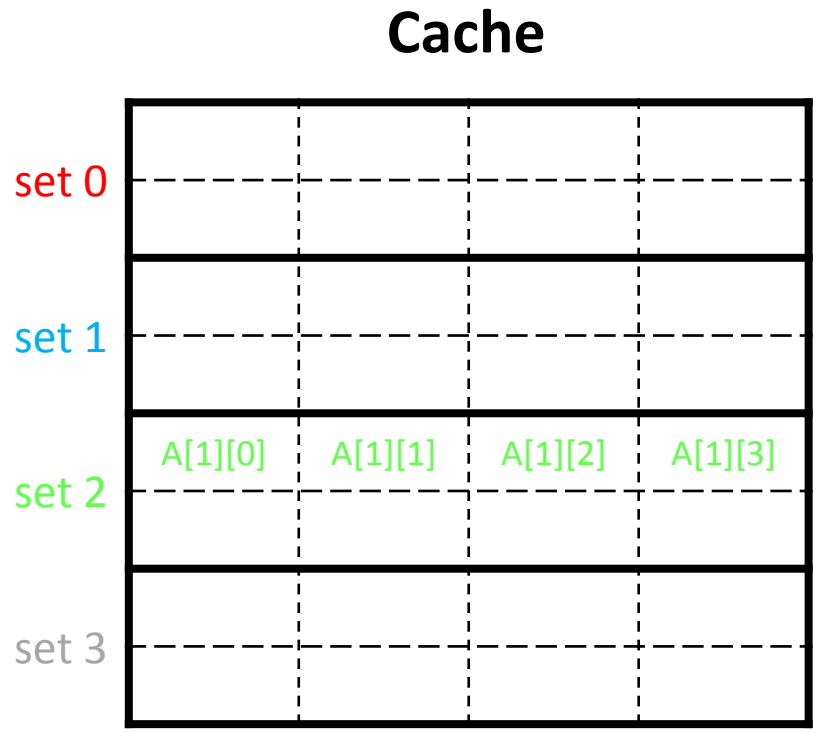
όλα στο set 0

A[0][0] → 0x0000 8000 = 0000 0000 0000 0000 0001 0000 0000 0000
B[0] → 0x0008 0200 = 0000 0000 0000 0000 0001 0010 0000 0000

A[0][0] και B[0] γίνονται mapped στο ίδιο block

Άσκηση 3η – Μέρος Β (i)

i=0, j=0 A[1][0] compulsory miss

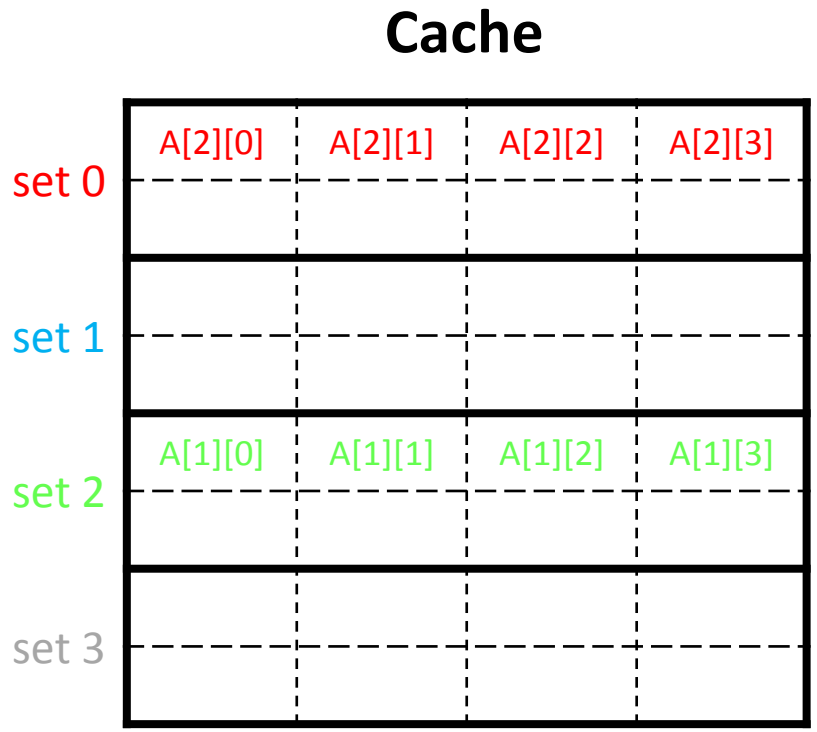


Άσκηση 3η – Μέρος Β (i)

i=0, j=0

A[1][0] compulsory miss

A[2][0] compulsory miss

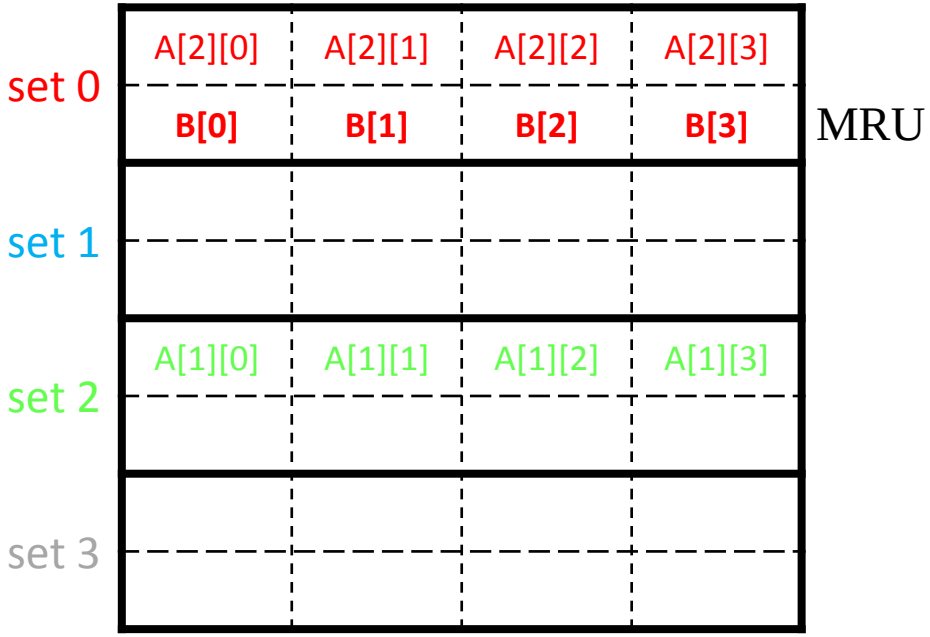


Άσκηση 3η – Μέρος B (i)

i=0, j=0

A[1][0]	compulsory miss
A[2][0]	compulsory miss
B[0]	compulsory miss

Cache



Άσκηση 3η – Μέρος B (i)

i=0, j=0

A[1][0]	compulsory miss
A[2][0]	compulsory miss
B[0]	compulsory miss
B[8]	compulsory miss

Cache

set 0	A[2][0]	A[2][1]	A[2][2]	A[2][3]
	B[0]	**B[1]**	**B[2]**	**B[3]**
set 1				
set 2	A[1][0]	A[1][1]	A[1][2]	A[1][3]
	B[8]	**B[9]**	**B[10]**	**B[11]**
set 3				

Άσκηση 3η – Μέρος B (i)

i=0, j=0

A[1][0]	compulsory miss
A[2][0]	compulsory miss
B[0]	compulsory miss
B[8]	compulsory miss
A[0][0]	compulsory miss

Cache

set 0	**A[0][0]**	**A[0][1]**	**A[0][2]**	**A[0][3]**
	B[0]	B[1]	B[2]	B[3]
set 1				
set 2	A[1][0]	A[1][1]	A[1][2]	A[1][3]
	B[8]	**B[9]**	**B[10]**	**B[11]**
set 3				

Άσκηση 3η – Μέρος B (i)

i=0, j=0	A[1][0]	compulsory miss
	A[2][0]	compulsory miss
	B[0]	compulsory miss
	B[8]	compulsory miss
	A[0][0]	compulsory miss
i=0, j=1	A[1][1]	hit

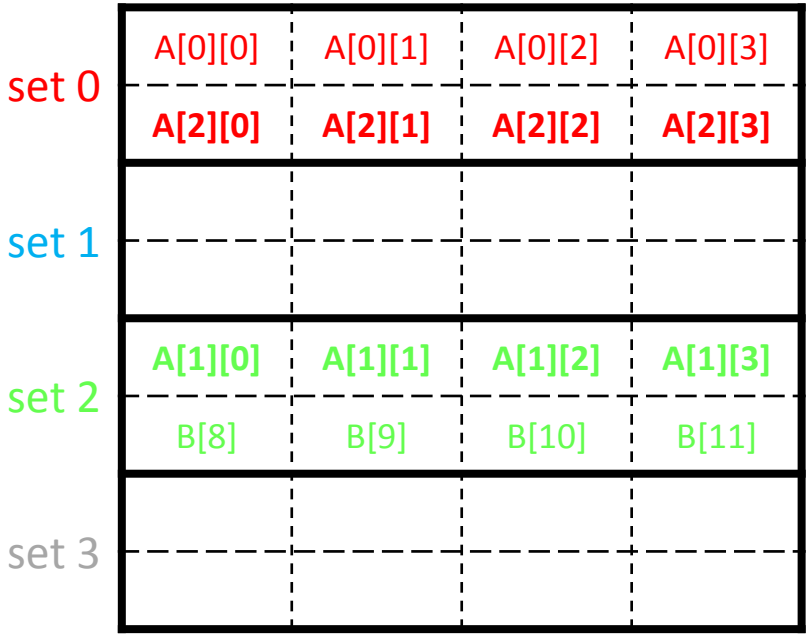
Cache

set 0	A[0][0]	A[0][1]	A[0][2]	A[0][3]
	B[0]	B[1]	B[2]	B[3]
set 1				
set 2	A[1][0]	A[1][1]	A[1][2]	A[1][3]
	B[8]	B[9]	B[10]	B[11]
set 3				

Άσκηση 3η – Μέρος B (i)

i=0, j=0	A[1][0]	compulsory miss
	A[2][0]	compulsory miss
	B[0]	compulsory miss
	B[8]	compulsory miss
	A[0][0]	compulsory miss
i=0, j=1	A[1][1]	hit
	A[2][1]	conflict miss

Cache



Άσκηση 3η – Μέρος B (i)

i=0, j=0	A[1][0]	compulsory miss
	A[2][0]	compulsory miss
	B[0]	compulsory miss
	B[8]	compulsory miss
	A[0][0]	compulsory miss
i=0, j=1	A[1][1]	hit
	A[2][1]	conflict miss
	B[1]	conflict miss

Cache

set 0	**B[0]**	**B[1]**	**B[2]**	**B[3]**
	A[2][0]	A[2][1]	A[2][2]	A[2][3]
set 1				
set 2	**A[1][0]**	**A[1][1]**	**A[1][2]**	**A[1][3]**
	B[8]	B[9]	B[10]	B[11]
set 3				

Άσκηση 3η – Μέρος Β (i)

i=0, j=0	A[1][0]	compulsory miss
	A[2][0]	compulsory miss
	B[0]	compulsory miss
	B[8]	compulsory miss
	A[0][0]	compulsory miss
i=0, j=1	A[1][1]	hit
	A[2][1]	conflict miss
	B[1]	conflict miss
	B[9]	hit

Cache

set 0	**B[0]**	**B[1]**	**B[2]**	**B[3]**
	A[2][0]	A[2][1]	A[2][2]	A[2][3]
set 1				
set 2	A[1][0]	A[1][1]	A[1][2]	A[1][3]
	B[8]	**B[9]**	**B[10]**	**B[11]**
set 3				

Άσκηση 3η – Μέρος Β (i)

i=0, j=0	A[1][0]	compulsory miss
	A[2][0]	compulsory miss
	B[0]	compulsory miss
	B[8]	compulsory miss
	A[0][0]	compulsory miss
i=0, j=1	A[1][1]	hit
	A[2][1]	conflict miss
	B[1]	conflict miss
	B[9]	hit
	A[0][1]	conflict miss

Cache

set 0	B[0]	B[1]	B[2]	B[3]
	A[0][0]	**A[0][1]**	**A[0][2]**	**A[0][3]**
set 1				
set 2	A[1][0]	A[1][1]	A[1][2]	A[1][3]
	B[8]	**B[9]**	**B[10]**	**B[11]**
set 3				

Άσκηση 3η – Μέρος Β (i)

i=0, j=0	A[1][0]	compulsory miss
	A[2][0]	compulsory miss
	B[0]	compulsory miss
	B[8]	compulsory miss
	A[0][0]	compulsory miss
i=0, j=1	A[1][1]	hit
	A[2][1]	conflict miss
	B[1]	conflict miss
	B[9]	hit
	A[0][1]	conflict miss

Cache

set 0	B[0]	B[1]	B[2]	B[3]
	A[0][0]	**A[0][1]**	**A[0][2]**	**A[0][3]**
set 1				
set 2	A[1][0]	A[1][1]	A[1][2]	A[1][3]
	B[8]	**B[9]**	**B[10]**	**B[11]**
set 3				

Για j=2, j=3 επαναλαμβάνεται το pattern του j=1

Άσκηση 3η – Μέρος Β (i)

i=0, j=4 A[1][4] compulsory miss

Cache

set 0	B[0]	B[1]	B[2]	B[3]
	A[0][0]	**A[0][1]**	**A[0][2]**	**A[0][3]**
set 1				
set 2	A[1][0]	A[1][1]	A[1][2]	A[1][3]
	B[8]	**B[9]**	**B[10]**	**B[11]**
set 3	A[1][4]	A[1][5]	A[1][6]	A[1][7]

Άσκηση 3η – Μέρος B (i)

i=0, j=4

A[1][4] compulsory miss

A[2][4] compulsory miss

Cache

set 0	B[0]	B[1]	B[2]	B[3]
	A[0][0]	**A[0][1]**	**A[0][2]**	**A[0][3]**
set 1	A[2][4]	A[2][5]	A[2][6]	A[2][7]
set 2	A[1][0]	A[1][1]	A[1][2]	A[1][3]
	B[8]	**B[9]**	**B[10]**	**B[11]**
set 3	A[1][4]	A[1][5]	A[1][6]	A[1][7]

Άσκηση 3η – Μέρος B (i)

i=0, j=4

A[1][4]	compulsory miss
A[2][4]	compulsory miss
B[4]	compulsory miss

Cache

set 0	B[0]	B[1]	B[2]	B[3]
	A[0][0]	**A[0][1]**	**A[0][2]**	**A[0][3]**
set 1	A[2][4]	A[2][5]	A[2][6]	A[2][7]
	B[4]	**B[5]**	**B[6]**	**B[7]**
set 2	A[1][0]	A[1][1]	A[1][2]	A[1][3]
	B[8]	**B[9]**	**B[10]**	**B[11]**
set 3	A[1][4]	A[1][5]	A[1][6]	A[1][7]

Άσκηση 3η – Μέρος Β (i)

i=0, j=4

A[1][4]	compulsory miss
A[2][4]	compulsory miss
B[4]	compulsory miss
B[12]	compulsory miss

Cache

set 0	B[0]	B[1]	B[2]	B[3]
	A[0][0]	**A[0][1]**	**A[0][2]**	**A[0][3]**
set 1	A[2][4]	A[2][5]	A[2][6]	A[2][7]
	B[4]	**B[5]**	**B[6]**	**B[7]**
set 2	A[1][0]	A[1][1]	A[1][2]	A[1][3]
	B[8]	**B[9]**	**B[10]**	**B[11]**
set 3	A[1][4]	A[1][5]	A[1][6]	A[1][7]
	B[12]	**B[13]**	**B[14]**	**B[15]**

Άσκηση 3η – Μέρος B (i)

i=0, j=4

A[1][4]	compulsory miss
A[2][4]	compulsory miss
B[4]	compulsory miss
B[12]	compulsory miss
A[0][4]	compulsory miss

Cache

set 0	B[0]	B[1]	B[2]	B[3]
	A[0][0]	**A[0][1]**	**A[0][2]**	**A[0][3]**
set 1	**A[0][4]**	**A[0][5]**	**A[0][6]**	**A[0][7]**
	B[4]	B[5]	B[6]	B[7]
set 2	A[1][0]	A[1][1]	A[1][2]	A[1][3]
	B[8]	**B[9]**	**B[10]**	**B[11]**
set 3	A[1][4]	A[1][5]	A[1][6]	A[1][7]
	B[12]	**B[13]**	**B[14]**	**B[15]**

Άσκηση 3η – Μέρος B (i)

i=0, j=4	A[1][4]	compulsory miss
	A[2][4]	compulsory miss
	B[4]	compulsory miss
	B[12]	compulsory miss
	A[0][4]	compulsory miss
i=0, j=5	A[1][5]	hit

Cache

set 0	B[0]	B[1]	B[2]	B[3]
	A[0][0]	**A[0][1]**	**A[0][2]**	**A[0][3]**
set 1	**A[0][4]**	**A[0][5]**	**A[0][6]**	**A[0][7]**
	B[4]	B[5]	B[6]	B[7]
set 2	A[1][0]	A[1][1]	A[1][2]	A[1][3]
	B[8]	**B[9]**	**B[10]**	**B[11]**
set 3	A[1][4]	A[1][5]	A[1][6]	A[1][7]
	B[12]	B[13]	B[14]	B[15]

Άσκηση 3η – Μέρος B (i)

i=0, j=4	A[1][4]	compulsory miss
	A[2][4]	compulsory miss
	B[4]	compulsory miss
	B[12]	compulsory miss
	A[0][4]	compulsory miss
i=0, j=5	A[1][5]	hit
	A[2][5]	conflict miss

Cache

set 0	B[0]	B[1]	B[2]	B[3]
	A[0][0]	**A[0][1]**	**A[0][2]**	**A[0][3]**
set 1	A[0][4]	A[0][5]	A[0][6]	A[0][7]
	A[2][4]	**A[2][5]**	**A[2][6]**	**A[2][7]**
set 2	A[1][0]	A[1][1]	A[1][2]	A[1][3]
	B[8]	**B[9]**	**B[10]**	**B[11]**
set 3	A[1][4]	A[1][5]	A[1][6]	A[1][7]
	B[12]	B[13]	B[14]	B[15]

Άσκηση 3η – Μέρος B (i)

i=0, j=4	A[1][4]	compulsory miss
	A[2][4]	compulsory miss
	B[4]	compulsory miss
	B[12]	compulsory miss
	A[0][4]	compulsory miss
i=0, j=5	A[1][5]	hit
	A[2][5]	conflict miss
	B[5]	conflict miss

Cache

set 0	B[0]	B[1]	B[2]	B[3]
	A[0][0]	**A[0][1]**	**A[0][2]**	**A[0][3]**
set 1	**B[5]**	**B[6]**	**B[7]**	**B[8]**
	A[2][4]	A[2][5]	A[2][6]	A[2][7]
set 2	A[1][0]	A[1][1]	A[1][2]	A[1][3]
	B[8]	**B[9]**	**B[10]**	**B[11]**
set 3	A[1][4]	A[1][5]	A[1][6]	A[1][7]
	B[12]	B[13]	B[14]	B[15]

Άσκηση 3η – Μέρος B (i)

i=0, j=4	A[1][4]	compulsory miss
	A[2][4]	compulsory miss
	B[4]	compulsory miss
	B[12]	compulsory miss
	A[0][4]	compulsory miss
i=0, j=5	A[1][5]	hit
	A[2][5]	conflict miss
	B[5]	conflict miss
	B[13]	hit

Cache

set 0	B[0]	B[1]	B[2]	B[3]
	A[0][0]	**A[0][1]**	**A[0][2]**	**A[0][3]**
set 1	**B[5]**	**B[6]**	**B[7]**	**B[8]**
	A[2][4]	A[2][5]	A[2][6]	A[2][7]
set 2	A[1][0]	A[1][1]	A[1][2]	A[1][3]
	B[8]	**B[9]**	**B[10]**	**B[11]**
set 3	A[1][4]	A[1][5]	A[1][6]	A[1][7]
	B[12]	**B[13]**	**B[14]**	**B[15]**

Άσκηση 3η – Μέρος Β (i)

i=0, j=4

A[1][4]	compulsory miss
A[2][4]	compulsory miss
B[4]	compulsory miss
B[12]	compulsory miss
A[0][4]	compulsory miss

i=0, j=5

A[1][5]	hit
A[2][5]	conflict miss
B[5]	conflict miss
B[13]	hit
A[0][5]	conflict miss

Cache

set 0	B[0]	B[1]	B[2]	B[3]
	A[0][0]	**A[0][1]**	**A[0][2]**	**A[0][3]**
set 1	B[5]	B[6]	B[7]	B[8]
	A[0][4]	**A[0][5]**	**A[0][6]**	**A[0][7]**
set 2	A[1][0]	A[1][1]	A[1][2]	A[1][3]
	B[8]	**B[9]**	**B[10]**	**B[11]**
set 3	A[1][4]	A[1][5]	A[1][6]	A[1][7]
	B[12]	**B[13]**	**B[14]**	**B[15]**

Άσκηση 3η – Μέρος B (i)

i=0, j=4	A[1][4]	compulsory miss
	A[2][4]	compulsory miss
	B[4]	compulsory miss
	B[12]	compulsory miss
	A[0][4]	compulsory miss
i=0, j=5	A[1][5]	hit
	A[2][5]	conflict miss
	B[5]	conflict miss
	B[13]	hit
	A[0][5]	conflict miss

Cache

set 0	B[0]	B[1]	B[2]	B[3]
	A[0][0]	**A[0][1]**	**A[0][2]**	**A[0][3]**
set 1	B[5]	B[6]	B[7]	B[8]
	A[0][4]	**A[0][5]**	**A[0][6]**	**A[0][7]**
set 2	A[1][0]	A[1][1]	A[1][2]	A[1][3]
	B[8]	**B[9]**	**B[10]**	**B[11]**
set 3	A[1][4]	A[1][5]	A[1][6]	A[1][7]
	B[12]	**B[13]**	**B[14]**	**B[15]**

Για j=6, j=7 επαναλαμβάνεται το pattern του j=5

Άσκηση 3η – Μέρος B (i)

i=1, j=0 A[2][0] conflict miss

Cache

set 0	**A[2][0]**	**A[2][1]**	**A[2][2]**	**A[2][3]**
	A[0][0]	A[0][1]	A[0][2]	A[0][3]
set 1	B[5]	B[6]	B[7]	B[8]
	A[0][4]	**A[0][5]**	**A[0][6]**	**A[0][7]**
set 2	A[1][0]	A[1][1]	A[1][2]	A[1][3]
	B[8]	**B[9]**	**B[10]**	**B[11]**
set 3	A[1][4]	A[1][5]	A[1][6]	A[1][7]
	B[12]	**B[13]**	**B[14]**	**B[15]**

Άσκηση 3η – Μέρος B (i)

i=1, j=0

A[2][0] conflict miss

A[3][0] compulsory miss

Cache

set 0	**A[2][0]**	**A[2][1]**	**A[2][2]**	**A[2][3]**
	A[0][0]	A[0][1]	A[0][2]	A[0][3]
set 1	B[5]	B[6]	B[7]	B[8]
	A[0][4]	**A[0][5]**	**A[0][6]**	**A[0][7]**
set 2	**A[3][0]**	**A[3][1]**	**A[3][2]**	**A[3][3]**
	B[8]	B[9]	B[10]	B[11]
set 3	A[1][4]	A[1][5]	A[1][6]	A[1][7]
	B[12]	**B[13]**	**B[14]**	**B[15]**

Άσκηση 3η – Μέρος B (i)

i=1, j=0

A[2][0]	conflict miss
A[3][0]	compulsory miss
B[0]	conflict miss

Cache

set 0	A[2][0]	A[2][1]	A[2][2]	A[2][3]
	B[0]	**B[1]**	**B[2]**	**B[3]**
set 1	B[5]	B[6]	B[7]	B[8]
	A[0][4]	**A[0][5]**	**A[0][6]**	**A[0][7]**
set 2	**A[3][0]**	**A[3][1]**	**A[3][2]**	**A[3][3]**
	B[8]	B[9]	B[10]	B[11]
set 3	A[1][4]	A[1][5]	A[1][6]	A[1][7]
	B[12]	**B[13]**	**B[14]**	**B[15]**

Άσκηση 3η – Μέρος B (i)

i=1, j=0

A[2][0]	conflict miss
A[3][0]	compulsory miss
B[0]	conflict miss
B[8]	hit

Cache

set 0	A[2][0]	A[2][1]	A[2][2]	A[2][3]
	B[0]	**B[1]**	**B[2]**	**B[3]**
set 1	B[5]	B[6]	B[7]	B[8]
	A[0][4]	**A[0][5]**	**A[0][6]**	**A[0][7]**
set 2	A[3][0]	A[3][1]	A[3][2]	A[3][3]
	B[8]	**B[9]**	**B[10]**	**B[11]**
set 3	A[1][4]	A[1][5]	A[1][6]	A[1][7]
	B[12]	**B[13]**	**B[14]**	**B[15]**

Άσκηση 3η – Μέρος Β (i)

i=1, j=0

A[2][0]	conflict miss
A[3][0]	compulsory miss
B[0]	conflict miss
B[8]	hit
A[1][0]	conflict miss

Cache

set 0	A[2][0]	A[2][1]	A[2][2]	A[2][3]
	B[0]	**B[1]**	**B[2]**	**B[3]**
set 1	B[5]	B[6]	B[7]	B[8]
	A[0][4]	**A[0][5]**	**A[0][6]**	**A[0][7]**
set 2	**A[1][0]**	**A[1][1]**	**A[1][2]**	**A[1][3]**
	B[8]	B[9]	B[10]	B[11]
set 3	A[1][4]	A[1][5]	A[1][6]	A[1][7]
	B[12]	**B[13]**	**B[14]**	**B[15]**

Άσκηση 3η – Μέρος Β (i)

i=1, j=0	A[2][0]	conflict miss
	A[3][0]	compulsory miss
	B[0]	conflict miss
	B[8]	hit
	A[1][0]	conflict miss
i=1, j=1	A[2][1]	hit

Cache

set 0	**A[2][0]**	**A[2][1]**	**A[2][2]**	**A[2][3]**
	B[0]	B[1]	B[2]	B[3]
set 1	B[5]	B[6]	B[7]	B[8]
	A[0][4]	**A[0][5]**	**A[0][6]**	**A[0][7]**
set 2	**A[1][0]**	**A[1][1]**	**A[1][2]**	**A[1][3]**
	B[8]	B[9]	B[10]	B[11]
set 3	A[1][4]	A[1][5]	A[1][6]	A[1][7]
	B[12]	**B[13]**	**B[14]**	**B[15]**

Άσκηση 3η – Μέρος Β (i)

i=1, j=0	A[2][0]	conflict miss
	A[3][0]	compulsory miss
	B[0]	conflict miss
	B[8]	hit
	A[1][0]	conflict miss
i=1, j=1	A[2][1]	hit
	A[3][1]	conflict miss

Cache

set 0	**A[2][0]**	**A[2][1]**	**A[2][2]**	**A[2][3]**
	B[0]	B[1]	B[2]	B[3]
set 1	B[5]	B[6]	B[7]	B[8]
	A[0][4]	**A[0][5]**	**A[0][6]**	**A[0][7]**
set 2	A[1][0]	A[1][1]	A[1][2]	A[1][3]
	A[3][0]	**A[3][1]**	**A[3][2]**	**A[3][3]**
set 3	A[1][4]	A[1][5]	A[1][6]	A[1][7]
	B[12]	**B[13]**	**B[14]**	**B[15]**

Άσκηση 3η – Μέρος B (i)

i=1, j=0	A[2][0]	conflict miss
	A[3][0]	compulsory miss
	B[0]	conflict miss
	B[8]	hit
	A[1][0]	conflict miss
i=1, j=1	A[2][1]	hit
	A[3][1]	conflict miss
	B[1]	hit

Cache

set 0	A[2][0]	A[2][1]	A[2][2]	A[2][3]
	B[0]	**B[1]**	**B[2]**	**B[3]**
set 1	B[5]	B[6]	B[7]	B[8]
	A[0][4]	**A[0][5]**	**A[0][6]**	**A[0][7]**
set 2	A[1][0]	A[1][1]	A[1][2]	A[1][3]
	A[3][0]	**A[3][1]**	**A[3][2]**	**A[3][3]**
set 3	A[1][4]	A[1][5]	A[1][6]	A[1][7]
	B[12]	**B[13]**	**B[14]**	**B[15]**

Άσκηση 3η – Μέρος B (i)

i=1, j=0	A[2][0]	conflict miss
	A[3][0]	compulsory miss
	B[0]	conflict miss
	B[8]	hit
	A[1][0]	conflict miss
i=1, j=1	A[2][1]	hit
	A[3][1]	conflict miss
	B[1]	hit
	B[9]	conflict miss

Cache

set 0	A[2][0]	A[2][1]	A[2][2]	A[2][3]
	B[0]	**B[1]**	**B[2]**	**B[3]**
set 1	B[5]	B[6]	B[7]	B[8]
	A[0][4]	**A[0][5]**	**A[0][6]**	**A[0][7]**
set 2	**B[8]**	**B[9]**	**B[10]**	**B[11]**
	A[3][0]	A[3][1]	A[3][2]	A[3][3]
set 3	A[1][4]	A[1][5]	A[1][6]	A[1][7]
	B[12]	**B[13]**	**B[14]**	**B[15]**

Άσκηση 3η – Μέρος B (i)

i=1, j=0	A[2][0]	conflict miss
	A[3][0]	compulsory miss
	B[0]	conflict miss
	B[8]	hit
	A[1][0]	conflict miss
i=1, j=1	A[2][1]	hit
	A[3][1]	conflict miss
	B[1]	hit
	B[9]	conflict miss
	A[1][1]	conflict miss

Cache

set 0	A[2][0]	A[2][1]	A[2][2]	A[2][3]
	B[0]	**B[1]**	**B[2]**	**B[3]**
set 1	B[5]	B[6]	B[7]	B[8]
	A[0][4]	**A[0][5]**	**A[0][6]**	**A[0][7]**
set 2	B[8]	B[9]	B[10]	B[11]
	A[1][0]	**A[1][1]**	**A[1][2]**	**A[1][3]**
set 3	A[1][4]	A[1][5]	A[1][6]	A[1][7]
	B[12]	**B[13]**	**B[14]**	**B[15]**

Άσκηση 3η – Μέρος Β (i)

i=1, j=0	A[2][0]	conflict miss
	A[3][0]	compulsory miss
	B[0]	conflict miss
	B[8]	hit
	A[1][0]	conflict miss
i=1, j=1	A[2][1]	hit
	A[3][1]	conflict miss
	B[1]	hit
	B[9]	conflict miss
	A[1][1]	conflict miss

Cache

set 0	A[2][0]	A[2][1]	A[2][2]	A[2][3]
	B[0]	**B[1]**	**B[2]**	**B[3]**
set 1	B[5]	B[6]	B[7]	B[8]
	A[0][4]	**A[0][5]**	**A[0][6]**	**A[0][7]**
set 2	B[8]	B[9]	B[10]	B[11]
	A[1][0]	**A[1][1]**	**A[1][2]**	**A[1][3]**
set 3	A[1][4]	A[1][5]	A[1][6]	A[1][7]
	B[12]	**B[13]**	**B[14]**	**B[15]**

Για j=2, j=3 επαναλαμβάνεται το pattern του j=1

Άσκηση 3η – Μέρος Β (i)

i=1, j=4 A[2][4] conflict miss

Cache

set 0	A[2][0]	A[2][1]	A[2][2]	A[2][3]
	B[0]	**B[1]**	**B[2]**	**B[3]**
set 1	**A[2][4]**	**A[2][5]**	**A[2][6]**	**A[2][7]**
	A[0][4]	A[0][5]	A[0][6]	A[0][7]
set 2	B[8]	B[9]	B[10]	B[11]
	A[1][0]	**A[1][1]**	**A[1][2]**	**A[1][3]**
set 3	A[1][4]	A[1][5]	A[1][6]	A[1][7]
	B[12]	**B[13]**	**B[14]**	**B[15]**

Άσκηση 3η – Μέρος Β (i)

i=1, j=4

A[2][4] conflict miss

A[3][4] compulsory miss

Cache

set 0	A[2][0]	A[2][1]	A[2][2]	A[2][3]
	B[0]	**B[1]**	**B[2]**	**B[3]**
set 1	**A[2][4]**	**A[2][5]**	**A[2][6]**	**A[2][7]**
	A[0][4]	A[0][5]	A[0][6]	A[0][7]
set 2	B[8]	B[9]	B[10]	B[11]
	A[1][0]	**A[1][1]**	**A[1][2]**	**A[1][3]**
set 3	A[3][4]	A[3][5]	A[3][6]	A[3][7]
	B[12]	B[13]	B[14]	B[15]

Άσκηση 3η – Μέρος B (i)

i=1, j=4

A[2][4]	conflict miss
A[3][4]	compulsory miss
B[4]	conflict miss

Cache

set 0	A[2][0]	A[2][1]	A[2][2]	A[2][3]
	B[0]	**B[1]**	**B[2]**	**B[3]**
set 1	A[2][4]	A[2][5]	A[2][6]	A[2][7]
	B[4]	**B[5]**	**B[6]**	**B[7]**
set 2	B[8]	B[9]	B[10]	B[11]
	A[1][0]	**A[1][1]**	**A[1][2]**	**A[1][3]**
set 3	A[3][4]	A[3][5]	A[3][6]	A[3][7]
	B[12]	B[13]	B[14]	B[15]

Άσκηση 3η – Μέρος Β (i)

i=1, j=4

A[2][4]	conflict miss
A[3][4]	compulsory miss
B[4]	conflict miss
B[12]	hit

Cache

set 0	A[2][0]	A[2][1]	A[2][2]	A[2][3]
	B[0]	**B[1]**	**B[2]**	**B[3]**
set 1	A[2][4]	A[2][5]	A[2][6]	A[2][7]
	B[4]	**B[5]**	**B[6]**	**B[7]**
set 2	B[8]	B[9]	B[10]	B[11]
	A[1][0]	**A[1][1]**	**A[1][2]**	**A[1][3]**
set 3	A[3][4]	A[3][5]	A[3][6]	A[3][7]
	B[12]	**B[13]**	**B[14]**	**B[15]**

Άσκηση 3η – Μέρος Β (i)

i=1, j=4

A[2][4]	conflict miss
A[3][4]	compulsory miss
B[4]	conflict miss
B[12]	hit
A[1][4]	conflict miss

Cache

set 0	A[2][0]	A[2][1]	A[2][2]	A[2][3]
	B[0]	**B[1]**	**B[2]**	**B[3]**
set 1	A[2][4]	A[2][5]	A[2][6]	A[2][7]
	B[4]	**B[5]**	**B[6]**	**B[7]**
set 2	B[8]	B[9]	B[10]	B[11]
	A[1][0]	**A[1][1]**	**A[1][2]**	**A[1][3]**
set 3	A[1][4]	A[1][5]	A[1][6]	A[1][7]
	B[12]	B[13]	B[14]	B[15]

Άσκηση 3η – Μέρος B (i)

i=1, j=4	A[2][4]	conflict miss
	A[3][4]	compulsory miss
	B[4]	conflict miss
	B[12]	hit
	A[1][4]	conflict miss
i=1, j=5	A[2][5]	hit

Cache

set 0	A[2][0]	A[2][1]	A[2][2]	A[2][3]
	B[0]	**B[1]**	**B[2]**	**B[3]**
set 1	**A[2][4]**	**A[2][5]**	**A[2][6]**	**A[2][7]**
	B[4]	B[5]	B[6]	B[7]
set 2	B[8]	B[9]	B[10]	B[11]
	A[1][0]	**A[1][1]**	**A[1][2]**	**A[1][3]**
set 3	A[1][4]	A[1][5]	A[1][6]	A[1][7]
	B[12]	B[13]	B[14]	B[15]

Άσκηση 3η – Μέρος B (i)

i=1, j=4	A[2][4]	conflict miss
	A[3][4]	compulsory miss
	B[4]	conflict miss
	B[12]	hit
	A[1][4]	conflict miss
i=1, j=5	A[2][5]	hit
	A[3][5]	conflict miss

Cache

set 0	A[2][0]	A[2][1]	A[2][2]	A[2][3]
	B[0]	**B[1]**	**B[2]**	**B[3]**
set 1	**A[2][4]**	**A[2][5]**	**A[2][6]**	**A[2][7]**
	B[4]	B[5]	B[6]	B[7]
set 2	B[8]	B[9]	B[10]	B[11]
	A[1][0]	**A[1][1]**	**A[1][2]**	**A[1][3]**
set 3	A[1][4]	A[1][5]	A[1][6]	A[1][7]
	A[3][4]	**A[3][5]**	**A[3][6]**	**A[3][7]**

Άσκηση 3η – Μέρος Β (i)

i=1, j=4	A[2][4]	conflict miss
	A[3][4]	compulsory miss
	B[4]	conflict miss
	B[12]	hit
	A[1][4]	conflict miss
i=1, j=5	A[2][5]	hit
	A[3][5]	conflict miss
	B[5]	hit

Cache

set 0	A[2][0]	A[2][1]	A[2][2]	A[2][3]
	B[0]	**B[1]**	**B[2]**	**B[3]**
set 1	A[2][4]	A[2][5]	A[2][6]	A[2][7]
	B[4]	**B[5]**	**B[6]**	**B[7]**
set 2	B[8]	B[9]	B[10]	B[11]
	A[1][0]	**A[1][1]**	**A[1][2]**	**A[1][3]**
set 3	A[1][4]	A[1][5]	A[1][6]	A[1][7]
	A[3][4]	**A[3][5]**	**A[3][6]**	**A[3][7]**

Άσκηση 3η – Μέρος Β (i)

i=1, j=4	A[2][4]	conflict miss
	A[3][4]	compulsory miss
	B[4]	conflict miss
	B[12]	hit
	A[1][4]	conflict miss
i=1, j=5	A[2][5]	hit
	A[3][5]	conflict miss
	B[5]	hit
	B[13]	conflict miss

Cache

set 0	A[2][0]	A[2][1]	A[2][2]	A[2][3]
	B[0]	**B[1]**	**B[2]**	**B[3]**
set 1	A[2][4]	A[2][5]	A[2][6]	A[2][7]
	B[4]	**B[5]**	**B[6]**	**B[7]**
set 2	B[8]	B[9]	B[10]	B[11]
	A[1][0]	**A[1][1]**	**A[1][2]**	**A[1][3]**
set 3	**B[12]**	**B[13]**	**B[14]**	**B[15]**
	A[3][4]	A[3][5]	A[3][6]	A[3][7]

Άσκηση 3η – Μέρος Β (i)

i=1, j=4	A[2][4]	conflict miss
	A[3][4]	compulsory miss
	B[4]	conflict miss
	B[12]	hit
	A[1][4]	conflict miss
i=1, j=5	A[2][5]	hit
	A[3][5]	conflict miss
	B[5]	hit
	B[13]	conflict miss
	A[1][5]	conflict miss

Cache

set 0	A[2][0]	A[2][1]	A[2][2]	A[2][3]
	B[0]	**B[1]**	**B[2]**	**B[3]**
set 1	A[2][4]	A[2][5]	A[2][6]	A[2][7]
	B[4]	**B[5]**	**B[6]**	**B[7]**
set 2	B[8]	B[9]	B[10]	B[11]
	A[1][0]	**A[1][1]**	**A[1][2]**	**A[1][3]**
set 3	B[12]	B[13]	B[14]	B[15]
	A[1][4]	**A[1][5]**	**A[1][6]**	**A[1][7]**

Άσκηση 3η – Μέρος Β (i)

i=1, j=4	A[2][4]	conflict miss
	A[3][4]	compulsory miss
	B[4]	conflict miss
	B[12]	hit
	A[1][4]	conflict miss
i=1, j=5	A[2][5]	hit
	A[3][5]	conflict miss
	B[5]	hit
	B[13]	conflict miss
	A[1][5]	conflict miss

Cache

set 0	A[2][0]	A[2][1]	A[2][2]	A[2][3]
	B[0]	**B[1]**	**B[2]**	**B[3]**
set 1	A[2][4]	A[2][5]	A[2][6]	A[2][7]
	B[4]	**B[5]**	**B[6]**	**B[7]**
set 2	B[8]	B[9]	B[10]	B[11]
	A[1][0]	**A[1][1]**	**A[1][2]**	**A[1][3]**
set 3	B[12]	B[13]	B[14]	B[15]
	A[1][4]	**A[1][5]**	**A[1][6]**	**A[1][7]**

Για j=6, j=7 επαναλαμβάνεται το pattern του j=5

Άσκηση 3η – Μέρος B (i)

i=2, j=0 A[3][0] conflict miss

Cache

set 0	A[2][0]	A[2][1]	A[2][2]	A[2][3]
	B[0]	**B[1]**	**B[2]**	**B[3]**
set 1	A[2][4]	A[2][5]	A[2][6]	A[2][7]
	B[4]	**B[5]**	**B[6]**	**B[7]**
set 2	**A[3][0]**	**A[3][1]**	**A[3][2]**	**A[3][3]**
	A[1][0]	A[1][1]	A[1][2]	A[1][3]
set 3	B[12]	B[13]	B[14]	B[15]
	A[1][4]	**A[1][5]**	**A[1][6]**	**A[1][7]**

Άσκηση 3η – Μέρος Β (i)

i=2, j=0

A[3][0] conflict miss

A[4][0] compulsory miss

Cache

set 0	**A[4][0]**	**A[4][1]**	**A[4][2]**	**A[4][3]**
	B[0]	B[1]	B[2]	B[3]
set 1	A[2][4]	A[2][5]	A[2][6]	A[2][7]
	B[4]	**B[5]**	**B[6]**	**B[7]**
set 2	**A[3][0]**	**A[3][1]**	**A[3][2]**	**A[3][3]**
	A[1][0]	A[1][1]	A[1][2]	A[1][3]
set 3	B[12]	B[13]	B[14]	B[15]
	A[1][4]	**A[1][5]**	**A[1][6]**	**A[1][7]**

Άσκηση 3η – Μέρος B (i)

i=2, j=0

A[3][0]	conflict miss
A[4][0]	compulsory miss
B[0]	hit

Cache

set 0	A[4][0]	A[4][1]	A[4][2]	A[4][3]
	B[0]	**B[1]**	**B[2]**	**B[3]**
set 1	A[2][4]	A[2][5]	A[2][6]	A[2][7]
	B[4]	**B[5]**	**B[6]**	**B[7]**
set 2	**A[3][0]**	**A[3][1]**	**A[3][2]**	**A[3][3]**
	A[1][0]	A[1][1]	A[1][2]	A[1][3]
set 3	B[12]	B[13]	B[14]	B[15]
	A[1][4]	**A[1][5]**	**A[1][6]**	**A[1][7]**

Άσκηση 3η – Μέρος Β (i)

i=2, j=0

A[3][0]	conflict miss
A[4][0]	compulsory miss
B[0]	hit
B[8]	conflict miss

Cache

set 0	A[4][0]	A[4][1]	A[4][2]	A[4][3]
	B[0]	**B[1]**	**B[2]**	**B[3]**
set 1	A[2][4]	A[2][5]	A[2][6]	A[2][7]
	B[4]	**B[5]**	**B[6]**	**B[7]**
set 2	A[3][0]	A[3][1]	A[3][2]	A[3][3]
	B[8]	**B[9]**	**B[10]**	**B[11]**
set 3	B[12]	B[13]	B[14]	B[15]
	A[1][4]	**A[1][5]**	**A[1][6]**	**A[1][7]**

Άσκηση 3η – Μέρος Β (i)

i=2, j=0

A[3][0]	conflict miss
A[4][0]	compulsory miss
B[0]	hit
B[8]	conflict miss
A[2][0]	conflict miss

Cache

set 0	**A[2][0]**	**A[2][1]**	**A[2][2]**	**A[2][3]**
	B[0]	B[1]	B[2]	B[3]
set 1	A[2][4]	A[2][5]	A[2][6]	A[2][7]
	B[4]	**B[5]**	**B[6]**	**B[7]**
set 2	A[3][0]	A[3][1]	A[3][2]	A[3][3]
	B[8]	**B[9]**	**B[10]**	**B[11]**
set 3	B[12]	B[13]	B[14]	B[15]
	A[1][4]	**A[1][5]**	**A[1][6]**	**A[1][7]**

Άσκηση 3η – Μέρος Β (i)

i=2, j=0	A[3][0]	conflict miss
	A[4][0]	compulsory miss
	B[0]	hit
	B[8]	conflict miss
	A[2][0]	conflict miss
i=2, j=1	A[3][1]	hit

Cache

set 0	**A[2][0]**	**A[2][1]**	**A[2][2]**	**A[2][3]**
	B[0]	B[1]	B[2]	B[3]
set 1	A[2][4]	A[2][5]	A[2][6]	A[2][7]
	B[4]	**B[5]**	**B[6]**	**B[7]**
set 2	**A[3][0]**	**A[3][1]**	**A[3][2]**	**A[3][3]**
	B[8]	B[9]	B[10]	B[11]
set 3	B[12]	B[13]	B[14]	B[15]
	A[1][4]	**A[1][5]**	**A[1][6]**	**A[1][7]**

Άσκηση 3η – Μέρος B (i)

i=2, j=0	A[3][0]	conflict miss
	A[4][0]	compulsory miss
	B[0]	hit
	B[8]	conflict miss
	A[2][0]	conflict miss
i=2, j=1	A[3][1]	hit
	A[4][1]	conflict miss

Cache

set 0	A[2][0]	A[2][1]	A[2][2]	A[2][3]
	A[4][0]	**A[4][1]**	**A[4][2]**	**A[4][3]**
set 1	A[2][4]	A[2][5]	A[2][6]	A[2][7]
	B[4]	**B[5]**	**B[6]**	**B[7]**
set 2	**A[3][0]**	**A[3][1]**	**A[3][2]**	**A[3][3]**
	B[8]	B[9]	B[10]	B[11]
set 3	B[12]	B[13]	B[14]	B[15]
	A[1][4]	A[1][5]	A[1][6]	A[1][7]

Άσκηση 3η – Μέρος B (i)

i=2, j=0

A[3][0]	conflict miss
A[4][0]	compulsory miss
B[0]	hit
B[8]	conflict miss
A[2][0]	conflict miss

i=2, j=1

A[3][1]	hit
A[4][1]	conflict miss
B[1]	conflict miss

Cache

set 0	**B[0]**	**B[1]**	**B[2]**	**B[3]**
	A[4][0]	A[4][1]	A[4][2]	A[4][3]
set 1	A[2][4]	A[2][5]	A[2][6]	A[2][7]
	B[4]	**B[5]**	**B[6]**	**B[7]**
set 2	**A[3][0]**	**A[3][1]**	**A[3][2]**	**A[3][3]**
	B[8]	B[9]	B[10]	B[11]
set 3	B[12]	B[13]	B[14]	B[15]
	A[1][4]	A[1][5]	A[1][6]	A[1][7]

Άσκηση 3η – Μέρος Β (i)

i=2, j=0

A[3][0]	conflict miss
A[4][0]	compulsory miss
B[0]	hit
B[8]	conflict miss
A[2][0]	conflict miss

i=2, j=1

A[3][1]	hit
A[4][1]	conflict miss
B[1]	conflict miss
B[9]	hit

Cache

set 0	**B[0]**	**B[1]**	**B[2]**	**B[3]**
	A[4][0]	A[4][1]	A[4][2]	A[4][3]
set 1	A[2][4]	A[2][5]	A[2][6]	A[2][7]
	B[4]	**B[5]**	**B[6]**	**B[7]**
set 2	A[3][0]	A[3][1]	A[3][2]	A[3][3]
	B[8]	**B[9]**	**B[10]**	**B[11]**
set 3	B[12]	B[13]	B[14]	B[15]
	A[1][4]	**A[1][5]**	**A[1][6]**	**A[1][7]**

Άσκηση 3η – Μέρος Β (i)

i=2, j=0

A[3][0]	conflict miss
A[4][0]	compulsory miss
B[0]	hit
B[8]	conflict miss
A[2][0]	conflict miss

i=2, j=1

A[3][1]	hit
A[4][1]	conflict miss
B[1]	conflict miss
B[9]	hit
A[2][1]	conflict miss

Cache

set 0	B[0]	B[1]	B[2]	B[3]
	A[2][0]	**A[2][1]**	**A[2][2]**	**A[2][3]**
set 1	A[2][4]	A[2][5]	A[2][6]	A[2][7]
	B[4]	**B[5]**	**B[6]**	**B[7]**
set 2	A[3][0]	A[3][1]	A[3][2]	A[3][3]
	B[8]	**B[9]**	**B[10]**	**B[11]**
set 3	B[12]	B[13]	B[14]	B[15]
	A[1][4]	**A[1][5]**	**A[1][6]**	**A[1][7]**

Άσκηση 3η – Μέρος Β (i)

i=2, j=0	A[3][0]	conflict miss
	A[4][0]	compulsory miss
	B[0]	hit
	B[8]	conflict miss
	A[2][0]	conflict miss
i=2, j=1	A[3][1]	hit
	A[4][1]	conflict miss
	B[1]	conflict miss
	B[9]	hit
	A[2][1]	conflict miss

Cache

set 0	B[0]	B[1]	B[2]	B[3]
	A[2][0]	**A[2][1]**	**A[2][2]**	**A[2][3]**
set 1	A[2][4]	A[2][5]	A[2][6]	A[2][7]
	B[4]	**B[5]**	**B[6]**	**B[7]**
set 2	A[3][0]	A[3][1]	A[3][2]	A[3][3]
	B[8]	**B[9]**	**B[10]**	**B[11]**
set 3	B[12]	B[13]	B[14]	B[15]
	A[1][4]	**A[1][5]**	**A[1][6]**	**A[1][7]**

Για j=2, j=3 επαναλαμβάνεται το pattern του j=1

Άσκηση 3η – Μέρος Β (i)

i=2, j=4	A[3][4]	conflict miss
	A[4][4]	compulsory miss
	B[4]	hit
	B[12]	conflict miss
	A[2][4]	conflict miss
i=2, j=5	A[3][5]	hit
	A[4][5]	conflict miss
	B[5]	conflict miss
	B[13]	hit
	A[2][5]	conflict miss

Cache

set 0	B[0]	B[1]	B[2]	B[3]
	A[2][0]	**A[2][1]**	**A[2][2]**	**A[2][3]**
set 1	**A[2][4]**	**A[2][5]**	**A[2][6]**	**A[2][7]**
	B[4]	B[5]	B[6]	B[7]
set 2	A[3][0]	A[3][1]	A[3][2]	A[3][3]
	B[8]	**B[9]**	**B[10]**	**B[11]**
set 3	**B[12]**	**B[13]**	**B[14]**	**B[15]**
	A[3][4]	A[3][5]	A[3][6]	A[3][7]

Για j=6, j=7 επαναλαμβάνεται το pattern του j=5

Άσκηση 3η – Μέρος Β (i)

Σύνολο για i = 0:

	αναφορές στη μνήμη				αποτέλεσμα				
i = 0:									
A[1][0]	A[2][0]	B[0]	B[8]	A[0][0]	m	m	m	m	m
A[1][1]	A[2][1]	B[1]	B[9]	A[0][1]	h	m	m	h	m
A[1][2]	A[2][2]	B[2]	B[10]	A[0][2]	h	m	m	h	m
A[1][3]	A[2][3]	B[3]	B[11]	A[0][3]	h	m	m	h	m
A[1][4]	A[2][4]	B[4]	B[12]	A[0][4]	m	m	m	m	m
A[1][5]	A[2][5]	B[5]	B[13]	A[0][5]	h	m	m	h	m
A[1][6]	A[2][6]	B[6]	B[14]	A[0][6]	h	m	m	h	m
A[1][7]	A[2][7]	B[7]	B[15]	A[0][7]	h	m	m	h	m

40 accesses
28 misses
12 hits

Άσκηση 3η – Μέρος Β (i)

Σύνολο για i = 1:

					αναφορές στη μνήμη			αποτέλεσμα	
i = 1:									
A[2][0]	A[3][0]	B[0]	B[8]	A[1][0]	m	m	m	h	m
A[2][1]	A[3][1]	B[1]	B[9]	A[1][1]	h	m	h	m	m
A[2][2]	A[3][2]	B[2]	B[10]	A[1][2]	h	m	h	m	m
A[2][3]	A[3][3]	B[3]	B[11]	A[1][3]	h	m	h	m	m
A[2][4]	A[3][4]	B[4]	B[12]	A[1][4]	m	m	m	h	m
A[2][5]	A[3][5]	B[5]	B[13]	A[1][5]	h	m	h	m	m
A[2][6]	A[3][6]	B[6]	B[14]	A[1][6]	h	m	h	m	m
A[2][7]	A[3][7]	B[7]	B[15]	A[1][7]	h	m	h	m	m

40 accesses
26 misses
14 hits

Το ίδιο pattern επαναλαμβάνεται για i=3

Άσκηση 3η – Μέρος Β (i)

Σύνολο για i = 2:

	αναφορές στη μνήμη				αποτέλεσμα				
i = 2:									
A[3][0]	A[4][0]	B[0]	B[8]	A[3][0]	m	m	h	m	m
A[3][1]	A[4][1]	B[1]	B[9]	A[3][1]	h	m	m	h	m
A[3][2]	A[4][2]	B[2]	B[10]	A[3][2]	h	m	m	h	m
A[3][3]	A[4][3]	B[3]	B[11]	A[3][3]	h	m	m	h	m
A[3][4]	A[4][4]	B[4]	B[12]	A[3][4]	m	m	h	m	m
A[3][5]	A[4][5]	B[5]	B[13]	A[3][5]	h	m	m	H	m
A[3][6]	A[4][6]	B[6]	B[14]	A[3][6]	h	m	m	H	m
A[3][7]	A[4][7]	B[7]	B[15]	A[3][7]	h	m	m	h	m

40 accesses
26 misses
14 hits

Το ίδιο pattern επαναλαμβάνεται για i=4

Άσκηση 3η – Μέρος Β (i)

Συνολικά

Accesses : 40 * 5 = 200
Misses : 28 + 4*26 = 132
Hits : 12 + 4*14 = 68

Άσκηση 3η – Μέρος Β (ii)

2ο Ζητούμενο: Αν η κρυφή μνήμη αντικατασταθεί με μία ίδια κρυφή μνήμη αλλά write-no-allocate, πώς θα επηρεαστούν τα hits και misses; Δικαιολογήστε την απάντησή σας δίνοντας όπως και πριν το συνολικό τους αριθμό.

Άσκηση 3η – Μέρος Β (ii)

Σύνολο για i = 0:

	αναφορές στη μνήμη				αποτέλεσμα				
i = 0:									
A[1][0]	A[2][0]	B[0]	B[8]	A[0][0]	m	m	m	m	m
A[1][1]	A[2][1]	B[1]	B[9]	A[0][1]	h	h	h	h	m
A[1][2]	A[2][2]	B[2]	B[10]	A[0][2]	h	h	h	h	m
A[1][3]	A[2][3]	B[3]	B[11]	A[0][3]	h	h	h	h	m
A[1][4]	A[2][4]	B[4]	B[12]	A[0][4]	m	m	m	m	m
A[1][5]	A[2][5]	B[5]	B[13]	A[0][5]	h	h	h	h	m
A[1][6]	A[2][6]	B[6]	B[14]	A[0][6]	h	h	h	h	m
A[1][7]	A[2][7]	B[7]	B[15]	A[0][7]	h	h	h	h	m

40 accesses
16 misses
24 hits

Άσκηση 3η – Μέρος Β (ii)

Σύνολο για i = 1:

	αναφορές στη μνήμη				αποτέλεσμα				
i = 1:									
A[2][0]	A[3][0]	B[0]	B[8]	A[1][0]	h	m	h	h	m
A[2][1]	A[3][1]	B[1]	B[9]	A[1][1]	h	h	h	h	m
A[2][2]	A[3][2]	B[2]	B[10]	A[1][2]	h	h	h	h	m
A[2][3]	A[3][3]	B[3]	B[11]	A[1][3]	h	h	h	h	m
A[2][4]	A[3][4]	B[4]	B[12]	A[1][4]	h	m	h	h	m
A[2][5]	A[3][5]	B[5]	B[13]	A[1][5]	h	h	h	h	m
A[2][6]	A[3][6]	B[6]	B[14]	A[1][6]	h	h	h	h	m
A[2][7]	A[3][7]	B[7]	B[15]	A[1][7]	h	h	h	h	m

40 accesses
10 misses
30 hits

Το ίδιο pattern επαναλαμβάνεται για i=3

Άσκηση 3η – Μέρος Β (ii)

Σύνολο για i = 2:

	αναφορές στη μνήμη				αποτέλεσμα				
i = 2:									
A[3][0]	A[4][0]	B[0]	B[8]	A[3][0]	h	m	h	h	m
A[3][1]	A[4][1]	B[1]	B[9]	A[3][1]	h	h	h	h	m
A[3][2]	A[4][2]	B[2]	B[10]	A[3][2]	h	h	h	h	m
A[3][3]	A[4][3]	B[3]	B[11]	A[3][3]	h	h	h	h	m
A[3][4]	A[4][4]	B[4]	B[12]	A[3][4]	h	m	h	h	m
A[3][5]	A[4][5]	B[5]	B[13]	A[3][5]	h	h	h	h	m
A[3][6]	A[4][6]	B[6]	B[14]	A[3][6]	h	h	h	h	m
A[3][7]	A[4][7]	B[7]	B[15]	A[3][7]	h	h	h	h	m

40 accesses
26 misses
14 hits

Το ίδιο pattern επαναλαμβάνεται για i=4

Άσκηση 3η – Μέρος Β (ii)

Συνολικά

Accesses : 40 * 5 = 200
Misses : 16 + 4*10 = 56
Hits : 24 + 4*30 = 144