Homework 3

20125071 – Bùi Lê Gia Cát

# Problem 1

## a/

The register %rdx is increased by 8 every step in the inner loop. In C, the size of the data type *long* is 8 bytes. As a result, at each step of the inner loop, the register percent %rdx is increased by 8 to move to the next element of A. Therefore, %rdx holds the pointer to element A[i][j].

## b/

In Assembly, there are instructions “movq (%rdx), %rcx” and “movq %rcx,(%rax)”. These instructions swap the address of the register %rdx and %rax. The register %rdx holds the address of A[i][j], so the register %rax must hold the address of A[j][i].

## c/

In Assembly, in line 7, there is an instruction “addq $120, %rax”. This instruction means changing the address of A[j][i] hold by the register %rax to the address of A[j+1][i]. So, the total size of M elements data types is 120. Hence M = 120/8 = 15.

# Problem 2

The instructions “addq $1,%rdx” and “cmpq %rdi,%rdx” are found on Assembly lines 13 and 15. In C, these instructions are equivalent to "i++" and “i < NR(n)”. As a result, the register %rdi keeps the result of NR(n). There are instructions “leaq (%rdi,%rdi,2), %rax” and “movq %rax, %rdi” on lines 3 and 4. Hence, NR(n) = 3x%rdi = 3n.

The register %rcx, which stores the address of element A[i][j], increases %r8 bytes to move to A[i+1][j], hence the register %r8 stores the size of NC(n) elements. The value in the register %r8 is shifted left by 3 or multiplied by 8 in Assembly line 7. Therefore, the value saved in the register %r8 in line 2 is the definition of NC(n) = %rdi\*4+1 = 4n+1

# Problem 3

## Ảnh có chứa bàn Mô tả được tạo tự độnga/

## b/

In C, *eval* passes the function *process* with a StrA s, while in Assembly, there is an instruction “leaq 64(%rsp),%rdi”. In Assembly, the register %rdi is use to pass the first parameter. Then, we know that %rsp+64 is the pointer to the begin of the first paprameter of function *process*. Therefore, *eval* passes %rsp+64 to function *process*.

## c/

Since *eval* passes %rsp+64 to function *process*, the Assembly code adds the register %rsp with an offset to access the elements in structure argument s.

## d/

Since *eval* passes %rsp+64 to the register %rdi, the function *process* calculate %rdi + offset to access and set value of the elements of structure r in *eval*.

## e/

## f/

In Assembly, the caller allocates stack space and passes the address to the callee, which then stores data in the space and returns the address to the caller.

# Problem 4

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Cache** | **m** | **C** | **B** | **E** | **S** | **t** | **s** | **b** |
| 1. | 32 | 1024 | 4 | 4 | 64 | 24 | 6 | 2 |
| 2. | 32 | 1024 | 4 | 256 | 1 | 30 | 0 | 2 |
| 3. | 32 | 1024 | 8 | 1 | 128 | 22 | 7 | 3 |
| 4. | 32 | 1024 | 8 | 128 | 1 | 29 | 0 | 3 |
| 5. | 32 | 1024 | 32 | 1 | 32 | 22 | 5 | 5 |
| 6. | 32 | 1024 | 32 | 4 | 8 | 24 | 3 | 5 |

# Problem 5

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Cache** | **m** | **C** | **B** | **E** | **S** | **t** | **s** | **b** |
| 1. | 32 | **2048** | 8 | 1 | **256** | 21 | 8 | 3 |
| 2. | 32 | 2048 | **4** | **4** | 128 | 23 | 7 | 2 |
| 3. | 32 | 1024 | 2 | 8 | 64 | **25** | **6** | 1 |
| 4. | 32 | 1024 | **32** | 2 | 16 | 23 | 4 | **5** |

# Problem 6

## a/

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CT | CT | CT | CT | CT | CT | CT | CT | CI | CI | CO | CO |
| 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

## b/

|  |  |  |  |
| --- | --- | --- | --- |
| **Operation** | **Address** | **Hit?** | **Read value (or unknow)** |
| Read | 0x834 | Miss | Unknown |
| Write | 0x836 | Hit |  |
| Read | 0xFFD | Hit | 0xC0 |

# Problem 7

The size of (C) of this cache in bytes is 128

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CT | CT | CT | CT | CT | CT | CT | CT | CI | CI | CI | CO | CO |
| 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

# Problem 8

Address format:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Memory reference:

|  |  |
| --- | --- |
| Parameter | Value |
| Blockoffset (CO) | 0x2 |
| Index (CI) | 0x6 |
| Cache tag (CT) | 0x38 |
| Cache hit? (Y/N) | Y |
| Cache byte return | 0xEB |

# Problem 9

## a/

Address format:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Memory reference:

|  |  |
| --- | --- |
| Parameter | Value |
| Blockoffset (CO) | 0x0 |
| Index (CI) | 0x2 |
| Cache tag (CT) | 0xB7 |
| Cache hit? (Y/N) | N |
| Cache byte return | ‒‒ |

## b/

* 0x1140, 0x1141, 0x1142, 0x1143
* 0x16C8, 0x16C9, 0x16CA, 0x16CB
* 0x180C, 0x180D, 0x180E, 0x180F
* 0x1150, 0x1151, 0x1152, 0x1153
* 0x1798, 0x1799, 0x179A, 0x179B

# Problem 10

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **dst array** | | | | |  | **src array** | | | | |
|  | **Col. 0** | **Col. 1** | **Col. 2** | **Col. 3** |  |  | **Col. 0** | **Col. 1** | **Col. 2** | **Col. 3** |
| **Row 0** | m | m | m | m |  | **Row 0 b** | m | h | h | h |
| **Row 1** | m | m | m | m |  | **Row 1** | m | h | h | h |
| **Row 2** | m | m | m | m |  | **Row 2** | m | h | h | h |
| **Row 3** | m | m | m | m |  | **Row 3** | m | h | h | h |