**CSC342**

**Solutions to Quiz No.2**

**Question 1 (30 Points)**

Memory model is a linear array of bytes as shown in Figure 1. The minimal addressable unit in this memory is one byte. Below depicts a small part of such a memory. The absolute address **0x80000100** is used as a base address and is stored in a register **RBase**. For clarity, this address is depicted to the left of the corresponding byte.

**-1**

**-1**

**-1**

**0x800000FD**

**0x800000FE**

**0x800000FF**

Addresses increase vertically from bottom to top

**+1**

**+1**

**+1**

**0x80000103**

**0x80000102**

**0x80000101**

Figure 1: Memory model is a linear array of bytes

**0x80000100**

|  |
| --- |
| **0x01** |
| **0x00** |
| **0x00** |
| **0x80** |
| **0x00** |
| **0x00** |
| **0x04** |

Q.1.1. [5 points] Assume you have a MIPS processor and associated memory as shown in Figure 1. What is the signed decimal value of the 32-Bit integer (word) at the address 0x80000100?

Explanation: Since MIPS is Big Endian, 0x80 is the most significant bit so it’s in decimal while 0x01 is the least significant bit so +1 in decimal. Hence,

Q.1.2. [5 points] Assume you have Intel i7 processor and associated memory as shown in Figure 1. What is the signed decimal value of the 32-Bit integer (word) at the address 0x80000100?

Explanation: Intel is little Endian, 0x00 is the most significant bit so it’s in decimal while 0x01 is the least significant bit so in decimal. Hence,

Q.1.3. [5 points] What is the address of a byte containing 0x01?

0x80000103

Explanation: See Purple

Q.1.4. [5 points] What is the offset from the base address (stored in Register RBase) to the byte containing 0x01?

Explanation: 0x80000103 – 0x80000100 = +3

Q.1.5. [5 points] What is the address of a byte containing 0x04?

0x800000FD

Explanation: See Orange

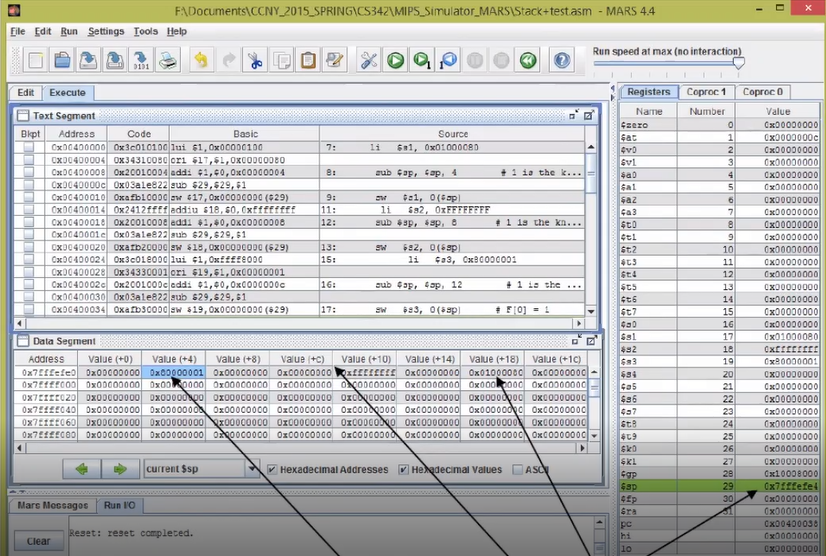
Q.1.6. [5 points] What is the offset from the base address (stored in Register RBase) to the byte containing 0x04?

-3

Explanation: 0x80000100 – 0x800000FD = -3

**Question 2 (25 Points)**

You are given an instance of a program in MARS MIPS simulator window



B

A

C

D

2.1. What is the signed decimal value of the integer on top of the stack?

0x80000001 =

Explanation: See A (Look at register $sp and go to data segment with corresponding address)

2.2. What is the value stored in the stack pointer register?

0x7FFFEFE4

Explanation: (See B) The stack pointer register is denoted by $sp

2.3.1 Compute the address of an integer stored on the stack at offset +12 from the stack pointer

0x7FFFEFF0

Explanation: 0x7FFFEFE4 + 0x0000000C => 0xE4+0x0C = F0

0xE4 1110 0100

+ 0x0C 0000 1100

------------------------------------

0xF0 1111 0000

2.3.2 What is the signed decimal value of the integer at this location?

Explanation: (See C) Apply two’s complement of 0xffffffff

2.4.1 Compute the address of an integer stored on the stack at offset +20 from the stack pointer

0x7FFFEFF8

Explanation: +20 = 0x14 :: 0x7FFFEFE4 + 0x00000014 => 0xE4 + 0x14

0xE4 1110 0100

+ 0x14 0001 0100

------------------------------------

0xF8 1111 1000

2.4.2. What is the signed decimal value of the integer at this location?

Explanation: (See D) 0x01000080 =

2.5 Can you determine the address of the instruction that will be executed next?

0x00400038

Explanation: Look in pc register

**Question 3. (32 Points)**

You are using MS Visual Studio development environment. The processor is Intel i7. In DEBUG mode you display the REGISTER, DISASSEMBLY, and MEMORY windows. Please answer the following questions based on the information displayed in the DEBUG mode windows.

1. (1 point) What is the content (what value is stored in EBP) of the base pointer register EBP?

0x006CF9E4

2. (1 point) Can you specify the Memory window # where partial stack frame is displayed? If yes, write YES and give the window number. If no, write NO.

Window #3

Explanation: Base pointer address (EBP) is displayed in that window

3. (10 points) Based on the information shown in the screenshots, can you determine if variable ***m*** is static or local? Please circle around your choice word. If it is possible, answer the following questions.

Explanation: It is local because in the Disassembly window, (39:36) (It is on the stack)

3.1. (2.5 points) What is the offset from the base pointer to local variable ***m*** on the stack?

0xD4 = 1101 0100 = -128 + 84 = -44

3.2. (2.5 points) Please list all *absolute addresses* of the offsets of variable ***m*** as used in

instructions the program:

0x00EB13D5, 0x00EB13DF

Explanation:

From mov instruction: 0x00EB13D3 + 0x02 = 0x00EB13D5

From sub instruction: 0x00EB13DD + 0x02 = 0x00EB13DF

3.3. (2.5 points) What is the address of local variable ***m*** on the stack?

0xE4 + 0xD4 = B8 0x006CF9B8

Explanation: Add offset to base pointer

0xE4 1110 0100

+ 0xD4 1101 0100

------------------------------------

0xB8 1011 1000

3.4. (2.5 points) What is the signed value (in DECIMAL) of local variable ***m*** as you can

observe on the *Stack Frame*?

-2

Explanation: (42:50) Using the address 0x006CF9B8 which contains ff ff ff fe and knowing it’s little Endian, apply two’s complement to 0xfffffffe to get -2

4. (10 points) Based on the information shown in the screenshots, can you determine if variable ***quizint*** is static or local?

4.1. (2.5 points) What is the offset from the base pointer to local variable ***quizint*** on the stack?

0xF8 = 1111 1000 = -8

Explanation: See machine instruction 0x00EB13BE in disassembly window

4.2. (2.5 points) Please list all absolute addresses of the offsets of variable ***quizint*** as used in

the program:

0x00EB13C0

Explanation: 0x00EB13BE + 0x02 = 0x00EB13C0

4.3. (2.5 points) What is the address of local variable ***quizint*** on the stack?

0xE4 + 0xF8 = DC 0x006CF9DC

Explanation: Add offset to base pointer

0xE4 1110 0100

+ 0xF8 1111 1000

------------------------------------

0xDC 1101 1100

4.4. (2.5 points) What is the signed value (in DECIMAL) of local variable ***quizint*** as you can

observe on the *Stack Frame*?

Explanation: Using the address 0x006CF9DC which contains the value 0x01000080 =

5. (10 points) Based on the information shown in the screenshots, can you determine if variable ***MIPSInt*** is static or local?

5.1. (2.5 points) What is the offset from the base pointer to local variable ***MIPSint*** on the stack?

0xE8 = 1110 1000 = -24

Explanation: See machine instruction 0x00EB13CC in disassembly window

5.2. (2.5 points) Please list all absolute addresses of the offsets of variable ***MIPSint*** as used in

the program:

0x00EB13CE

Explanation: 0x00EB13CC + 0x02 = 0x00EB13CE

5.3. (2.5 points) What is the address of local variable ***MIPSint*** on the stack?

0xE4 + 0xE8 = C4 0x006CF9C4

Explanation: Add offset to base pointer

0xE4 1110 0100

+ 0xE8 1110 1000

------------------------------------

0xCC 1100 1100

5.4. (2.5 points) What is the signed value (in DECIMAL) of local variable ***MIPSint*** as you can

observe on the *Stack Frame*?

0x80000001 =

Explanation: Using the address 0x006CF9C4 which contains the value 0x80000001 =

6. (1 point) Can you determine the address of the instruction that will be executed next instance?

0x00EB13DD

Explanation: Look at the program counter (i.e. EIP register value)

7. (1 point) What is the machine code length in bytes?

0xEB - 0xA0 + 0x01 = 0x4B + 0x01 = 0x4C = 76

0xEB 1110 1011

- 0xA0 1010 0000

+ 0110 0000 <= Two’s Complement

------------------------------------

0x4B 1000 1011

Explanation: (Last instruction) – (First instruction) + 1

8. (1 point) Can you determine the number of instructions of length 7 bytes? If yes, what is it?

There are 4 instructions of length 7 bytes: 00EB13BE, 00EB13C5, 00EB13CC, 00EB13D3.

Explanation: See the Disassembly window

9. (1 point) Can you determine the number of instructions of length 6 bytes? If yes, what is it?

There are 2 instructions of length 6 bytes: 00EB13A3 and 00EB13AC.

Explanation: See the Disassembly window

10. (1 point) Can you determine the number of instructions of length 5 bytes? If yes, what is it?

There are 2 instructions of length 5 bytes: 00EB13B2 and 00EB13B7.

Explanation: See the Disassembly window