HW #1

1615051 Young-Eun Lee

HW #1

1615051 Young-Eun Lee

Part 1

**# 1.3**

We have many high level language. And we have many word using alphabet, selection using 0 and 1. Computer is operated our order. We call this order “instruction”. And we create “assembler”. Assembler is program that is change the symbol expression to binary expression.

Transformation method is that instruction enter to computer. Then, assembler is change this instruction to binary expression. This binary expression meaning is same instruction meaning. In this assembler, we call “assembly language”. And Changing the high-level programming language to machine language call compiler.

**1.4**

a. One frame has (1280 X 1024) pixel. Method that how to get ‘the minimum size in bytes of the frame buffer to store a frame’ is that multiplication pixel and byte. We know each pixel have 3 color. And each color have 8 bit, in other words each pixel have 3 bytes(=24 bits).

As a result, (1280 X 1024 X 3 bytes) = 3,932,160 bytes/frame.

b. In the question a, we find the size of one frame in byte. Than we will change byte to bit. One frame is 3,932,160\*8=31,457,280 bit/frame. And 100Mbit is 10^8 bit. The time is 31,457,280/10^8 bit = 0.3145728 sec because the network sends 100 Mbit per second.

**2.1**

main: #start execution at main

li $v0, 5 #load syscall read\_int 5 $v0.

syscall #make the syscall

move $s0, $v0 #move the number read into $s0

sub $t0, $a0, $s0 #compute the subtraction of $a0 and $s0, and put it into $t0

add $t1, $a1, $t0 #compute the sum of $a1 and $t0, and put it into $t1

syscall #make the syscall

**2.2**

main: #start execution at main

add $t0, $s0, $s1 #compute the sum of $s0 and $s1, and put it into $t0

add $t1, $s2, $t0 #compute the sum of $s2 and $t0, and put it into $t1

syscall #make the syscall

Part 2

1. Code

|  |
| --- |
| .text  main:  la $a0, my\_msg0 #load the address of a my\_msg0 into $a0  li $v0, 4 #4 is the print\_string syscall.  syscall #make the syscall    la $a0, my\_msg1 #load the address of a my\_msg1 into $a0  li $v0, 4 #4 is the print\_string syscall.  syscall #make the syscall    la $a0, my\_msg2 #load the address of a my\_msg2 into $a0  li $v0, 4 #4 is the print\_string syscall.  syscall #make the syscall    la $a0, my\_msg3 #load the address of a my\_msg3 into $a0  li $v0, 4 #4 is the print\_string syscall.  syscall #make the syscall    la $a0, my\_msg4 #load the address of a my\_msg4 into $a0  li $v0, 4 #4 is the print\_string syscall.  syscall #make the syscall    la $a0, my\_msg5 #load the address of a my\_msg5 into $a0  li $v0, 4 #4 is the print\_string syscall.  syscall #make the syscall    la $a0, my\_msg6 #load the address of a my\_msg6 into $a0  li $v0, 4 #4 is the print\_string syscall.  syscall #make the syscall    li $v0, 10 #4 is the print\_string syscall.  syscall #make the syscall    .data  my\_msg0: .asciiz "Student ID 1615051: Young-Eun Lee\n" #the string "Student ID 1615051: Young-Eun Lee"  my\_msg1: .ascii "I want to learn" #The string "I want to learn"  .ascii " " #the space  .ascii "about the computer architecture.\n" #the string "about the computer architecture."  .byte 0 #a 0 byte.  my\_msg2: .ascii "I don't know my future grade" #the string "I don't know my future grade"  .ascii "and " #the word "and"  .ascii "I'm not confident to get the high grade.\n" #the string "I'm not confident to get the high grade."  .byte 0 #a 0 byte  my\_msg3: .asciiz "Because my credit is very low..\n" #the string "Because my credit is very low."  my\_msg4: .byte 0x42 #hex for ASCII "B"  .byte 0x75 #hex for ASCII "u"  .byte 0x74 #hex for ASCII "t"  .byte 0x20 #hex for ASCII space  .byte 0x0 #hex for ASCII NUL  my\_msg5: .asciiz "I will try to get good information about computer architecture.\n"  #the string "I will try to get good information about computer architecture."  my\_msg6: .ascii "Thanks you professor" #the string "Thanks you professor"  .ascii " " #the space.  .ascii "and teaching assistant.\n" #the string "and teaching assistant."  .byte 0 #a 0 byte. |

2. Capture the console output

