2차 실험보고서

실습명: 동기 순서 논리 회로 시뮬레이션과 분석,

MARS 도구를 사용한 어셈블리 샘플 프로그램 분석

I. 동기 순서논리 회로 (디지털 시계)

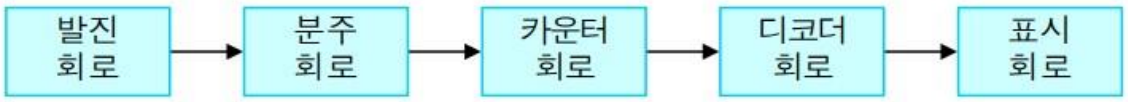
1. 실습목적

디지털 시계를 모의실험으로 구현하여 동작을 분석하여, 동기식 순서 논리 회로를 이해한다. 시, 분, 초에 사용된 2진, 6진, 10진 카운터의 동작 원리를 이해하고 디지털 시계의 원리를 알아본다.

2. 기본이론

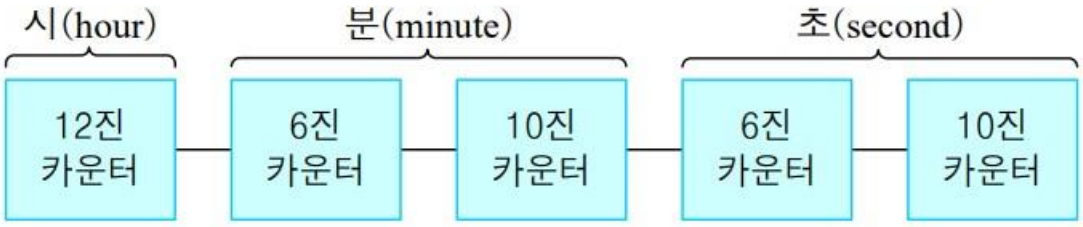
동기 순서회로에서 상태(state)는 클록펄스가 들어오는 시점에서 상태가 변화하는 회로이다. 클록펄스에 의해서 동작하는 회로를 동기순서논리회로라 한다.

발진 회로는 디지털 시계에 안정적인 클록을 제공할 목적으로 설계되는 회로이다. 분주 회로는 발진 회로로부터 얻어진 구형파를 이용해 디지털 시계의 기본단위인 1초를 나타내기 위한 1Hz 주파수를 얻는 회로이다. 카운터 회로의 출력인 2진 데이터를 표시하기 위해 디코터 회로 및 표시회로가 필요하다. 디코터 회로는 7447, 표시회로는 7세그먼트가 사용된다. 7447은 BCD-to-7세그먼트으로 BCD를 숫자로 표시해주는 소자이다.



<디지털 시계의 블록 다이어그램>

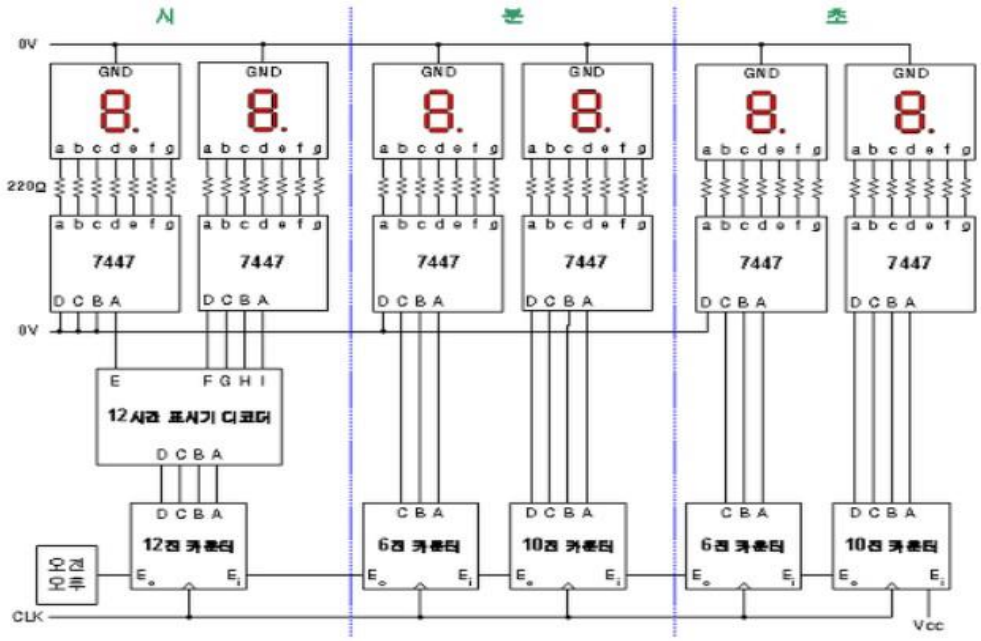
디지털 시계는 시간 표시를 위한 7세그먼트, BCD 카운터, modulo-m 카운터 등이 필요하다. Modulo-N 카운터의 종류는 6진, 10진, 12진 카운터가 있다.



<카운터 회로의 블록도>

3. 시뮬레이션 과정

초는 BCD 카운터에서 10진 카운터를 한다. 10진 카운터는 0~9까지 반복한다. 이 신호는 6진 카운터에서 6분주된 신호가 발생되므로 초에서 60분주된 신호인 1분의 신호가 발생된다. 즉, 이 카운터가 9까지 센 후 다시 0으로 돌아갈 때에 초의 십의자리에 해당하는 6진 카운터가 1 증가해야 한다. 분도 초와 동일한 구조로 되어 있어 1분에 대한 60분주 신호인 1시간 신호가 발생된다. 분의 일의 자리에 해당하는 10진 카운터는 초의 십의 자리에 해당하는 6진 카운터가 5에서 0으로 변하는 시점에 맞추어 1씩 증가해야 한다. 시는 12시간 주기 12진 카운터를 사용해 신호를 발생시킨다.



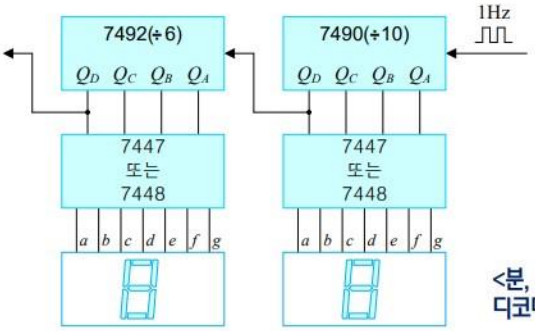
- 초의 일의자리 10진 카운터는 클럭이 인가될 때마다 증가된다.

- 초의 십의자리 6진 카운터는 초의 일의 자리 10진 카운터가 9에서 0으로 변할 때 증가된다.

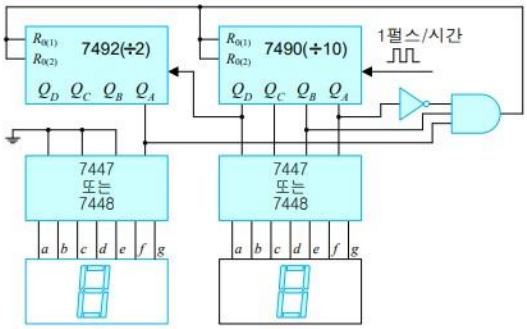
- 분의 일의 자리 10진 카운터는 초의 십의 자리 6진 카운터가 5에서 0으로 변할 때 (59초에서 00초로 변할 때) 증가된다.

- 분의 십의 자리 6진카운터는 분의 일의 자리 10진 카운터가 9에서 0으로 변할 때 (x9분 59초에서 x9분 00초로 변할 때) 증가된다.

- 시를 나타내는 12진 카운터는 분의 십의 자리 6진 카운터가 5에서 0으로 변할 때 (59분 59초에서 00분 00초로 변할 때) 증가된다.

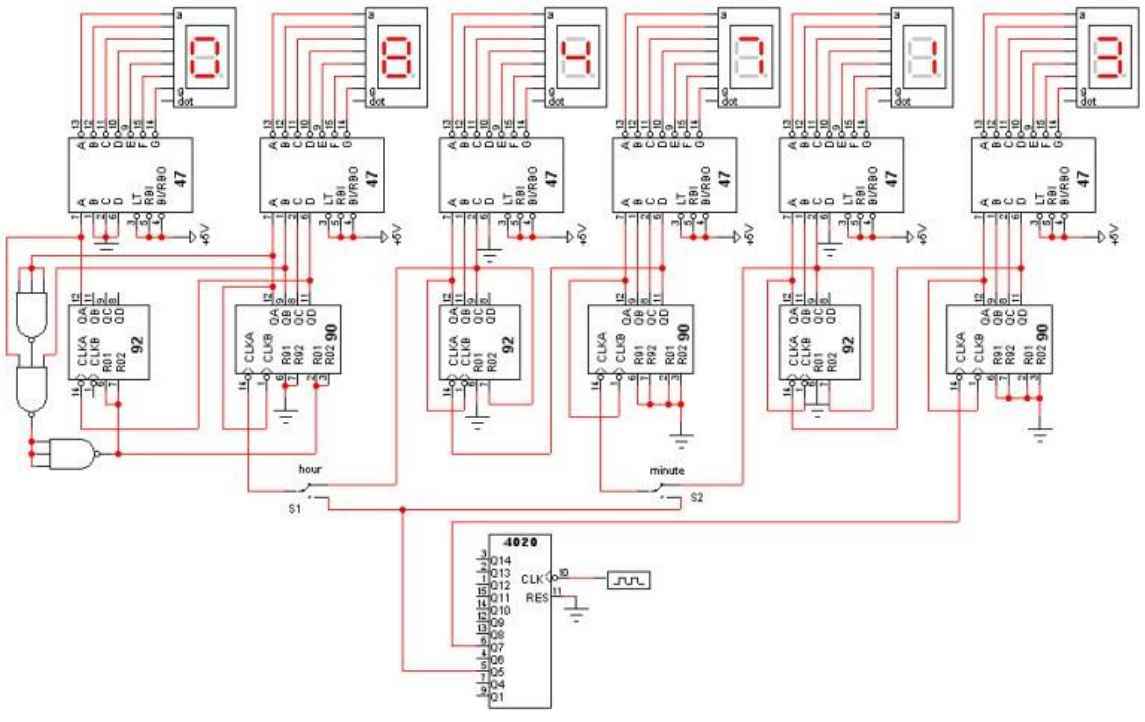


<분, 초 단위의 카운터
디코더 및 드라이브 회로>

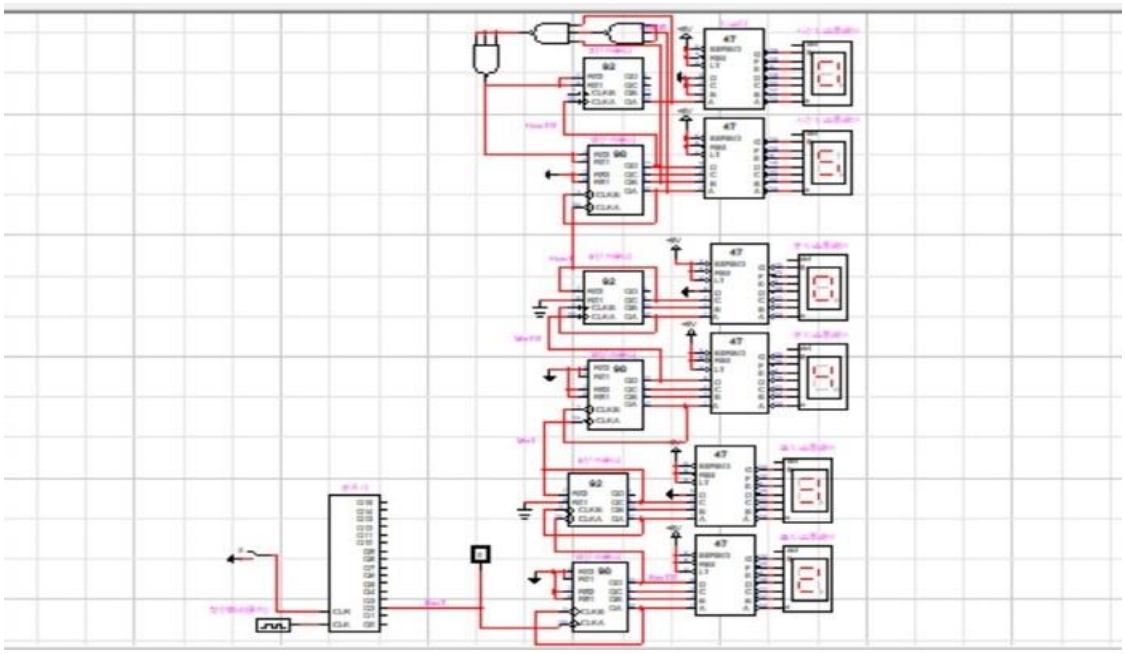


<시 단위의 카운터,
디코더 및 드라이브 회로>

2진 카운터와 6진 카운터는 7492에 독립적으로 내장되어 있다. 2진 카운터와 5진 카운터는 7490에 독립적으로 내장되어 있다. 7490과 7492의 2진 카운터 데이터 값을 BCD 출력 7447핀을 사용하여 입력으로 넣어 Display를 구동한다. BCD 코드를 해석해 10진수 표현으로 나타내기 위한 것이다.

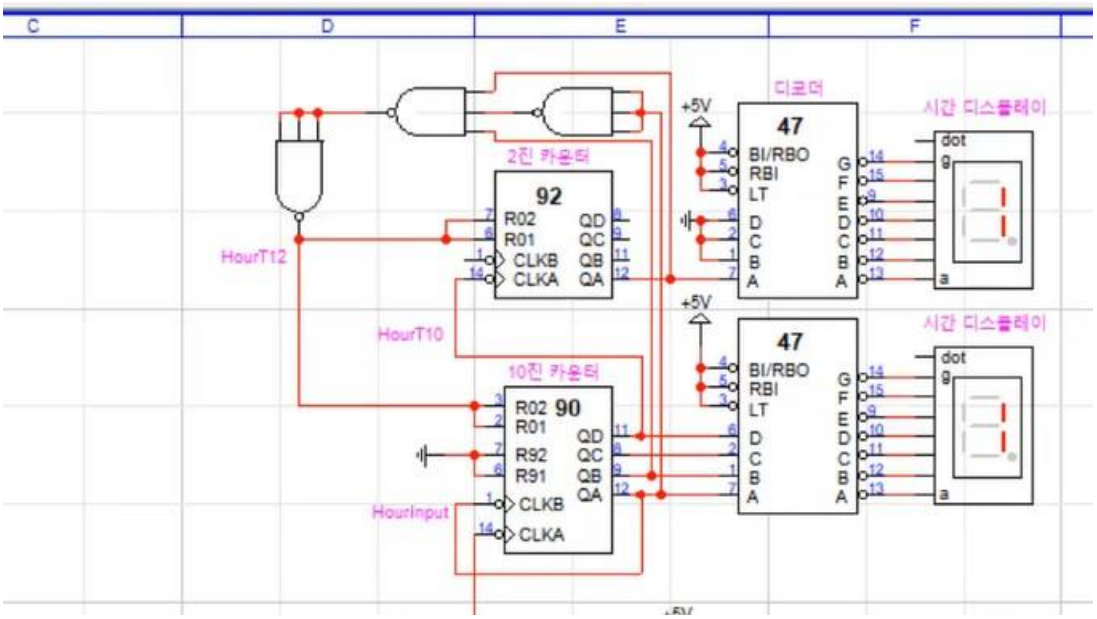


4. 실험결과와 분석



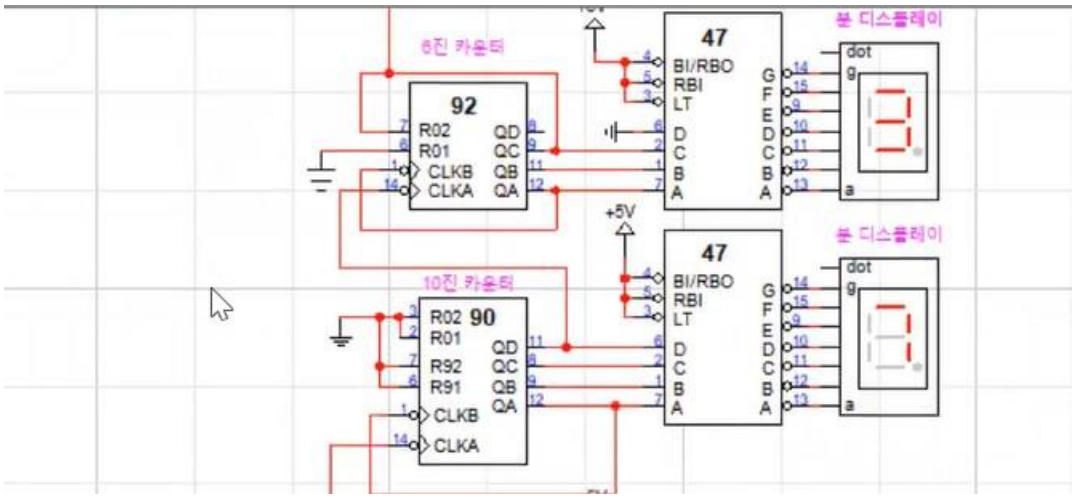
시-실험결과

시(hour)에는 12 진 카운터(7490)가 사용된다. 2 진 카운터는 10 진 카운터가 9 에서 0 이 되는 순간 1 증가한다. 그 다음 과정에는 2 진 카운터는 1 에서 0 으로 돌아간다. 시의 일의 자리 10 진 카운터는 분의 십의 자리 6 진 카운터가 5 에서 0 으로 변하는 시점에 맞추어 1 씩 증가해야 한다. 디코더를 사용해 7490,92 의 2 진수를 입력 받아 BCD 로 변환하여 입력으로 받아 직접 7 세그먼트에 출력을 만들어낸다.



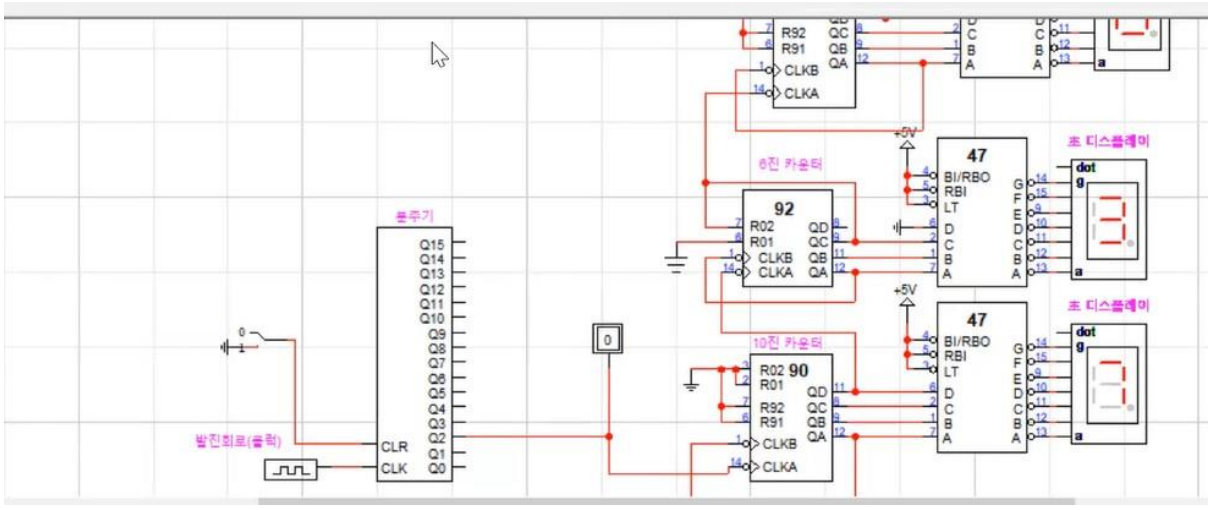
분-실험결과

분의 일의 자리 10 진 카운터는 초의 십의 자리 6 진 카운터가 5 에서 0 으로 변하는 시점에 맞추어 1 씩 증가해야 한다. 또한 10 진 카운터가 9 에서 0 이 될 때 6 진 카운터는 1 씩 증가한다. 디코더를 사용해 7490, 92 의 2 진수를 입력 받아 BCD 로 변환하여 입력으로 받아 직접 7 세그먼트에 출력을 만들어낸다.



초-실험결과

초의 일의 자리 10 진 카운터는 0~9 까지 세는 동안 다른 카운터들은 동작하면 안 되고 9 까지 센 후 다시 0 으로 돌아갈 때에 초의 십의 자리 6 진 카운터가 1 씩 증가해야 한다.



이 실험 결과를 통해 시계를 구성하기 위해 2진, 6진, 12진 카운터가 필요하며 클럭 펄스가 하나씩 인가될 때마다 각 카운터의 1씩이 증가하는 시점이 다르다는 것을 알게 되었다. 또한 7490,92의 결과인 2진 데이터를 디코터 회로 7447의 입력으로 넣어 BCD 코드를 해석해 10진수 표현을 Display에 나타낸다는 것을 알게 되었다.

II. MARS 도구를 사용한 어셈블리 샘플 프로그램 분석

1. 실습목적

MARS를 사용해 간단한 어셈블리 샘플 프로그램이 동작하면서 CPU 내부 레지스터 상태가 변화 상태를 파악해 CPU 동작원리를 이해한다. 실습을 통해 명령어에 대해 배우고 코드가 실행되는 과정을 이해한다.

2. 기본이론

중앙 처리 장치(CPU) 자체에 존재하는 첫 번째 범주인 메모리는 레지스터라고 한다. 레지스터 메모리는 매우 제한적이며 종종 CPU의 레지스터 파일이라고 하는 것에 포함된다. 이 유형의 메모리는 이 텍스트에서 레지스터라고 한다.

레지스터는 CPU에 존재하는 제한된 양의 메모리이다. 레지스터에 저장되지 않은 데이터는 CPU에서 작동할 수 없다. CPU가 데이터를 사용하려면 먼저 메모리, 사용자 또는 디스크 드라이브의 데이터를 레지스터에 로드해야 한다. MIPS CPU에는 32개의 레지스터만 있으며 각 레지스터는 단일 32비트 값을 저장하는 데 사용할 수 있다.

메모리의 데이터를 사용하기 위해서는 로드/저장 명령을 사용하여 읽고 쓸 주소와 데이터를 시스템 버스에 놓고 메모리에서 CPU로 전송한다. 데이터 및 주소는 일반적으로 로드 워드(lw) 또는 스토어 워드(sw) 작업을 사용하여 시스템 버스에 배치된다. 그런 다음 데이터는 메모리에서 레지스터로 읽거나 쓴다. 프로그램에서 32개 이상의 데이터 값을 사용하려면 해당 값이 메모리에 존재해야 하며 레지스터에 로드하여 사용해야 한다.

- **레지스터**

Mnemonic	Number	.....	Mnemonic	Number	.....	Mnemonic	Number
$zero	$0		$t3	$11		$s6	$22
$at	$1		$t4	$12		$s7	$23
$v0	$2		$t5	$13		$t8	$24
$v1	$3		$t6	$14		$t9	$25
$a0	$4		$t7	$15		$k0	$26
$a1	$5		$s0	$16		$k1	$27
$a2	$6		$s1	$17		$gp	$28
$a3	$7		$s2	$18		$sp	$29
$t0	$8		$s3	$19		$fp	$30
$t1	$9		$s4	$20		$ra	$31
$t2	$10		$s5	$21			

Table 2-1: Register Conventions

- $zero ($0) - 항상 상수 값 0을 포함하는 특수 목적 레지스터. 읽을 수는 있지만 쓸 수는 없다.
- $at ($1) - 어셈블러용으로 예약된 레지스터. 어셈블러가 임시 레지스터를 사용해야 하는 경우 $at를 사용하므로 이 레지스터는 프로그래머용으로 사용할 수 없다.
- $v0-$v1 ($2-$3) - 레지스터는 일반적으로 하위 프로그램의 반환 값에 사용된다. $v0은 요청된 서비스를 syscall에 입력하는 데에도 사용된다.
- $a0-$a3 ($4-$7) - 레지스터는 인수(또는 매개변수)를 하위 프로그램으로 전달하는 데 사용된다.
- $t0-$t9 ($8-$15, $24-$25) - 레지스터는 임시 변수를 저장하는 데 사용된다. 임시 변수의 값은 서브프로그램이 호출될 때 변경될 수 있다.
- $s0-$s8 ($16-$24) - 레지스터는 저장된 값을 저장하는 데 사용된다. 이러한 레지스터의 값은 서브프로그램 호출 간에 유지된다.
- $k0-$k1 ($26-$27) - 레지스터는 운영 체제에서 사용되며 프로그래머용으로는 사용할 수 없다.

- $gp ($28) - 전역 메모리에 대한 포인터. 힙 할당과 함께 사용된다.
- $sp ($29) - 스택에서 이 메서드에 대한 데이터의 시작 부분을 추적하는 데 사용되는 스택 포인터.
- $fp ($30) - 스택에 대한 정보를 유지하기 위해 $sp와 함께 사용되는 프레임 포인터. 이 텍스트는 메서드 호출에 $fp를 사용하지 않는다.
- $ra ($31) - 반환 주소: 하위 프로그램에서 반환할 때 사용할 주소에 대한 포인터.

-아래 그림은 MIPS 컴퓨터에서 4G 메모리가 구성되는 방식을 보여준다. **MIPS 에서 사용하는 메모리 유형**은 다음과 같다.

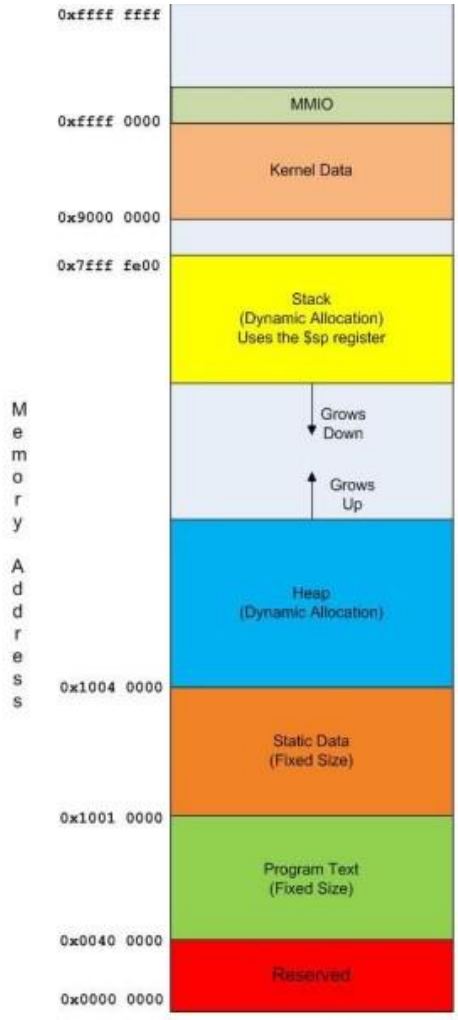


Figure 2-3: MIPS memory configuration

- Reserved- MIPS 플랫폼용으로 예약된 메모리다. 이 주소의 메모리는 프로그램에서 사용할 수 없다.
- Program text- (주소 0x0040 0000 - 0x1000 00000) 프로그램의 기계 코드 표현이 저장되는 곳이다. 각 명령어는 이 메모리에 워드(32비트 또는 4바이트)로 저장된다. 모든 명령은 4의 배수(0x0040 0000, 0x0040 0004, 0x0040 0080, 0x0040 00B0 등)인 단어 경계에 속한다.
- Static data - (주소 0x1001 0000 - 0x1004 0000) 프로그램의 데이터 세그먼트에서 오는 데이터다. 섹션의 요소 크기는 프로그램 생성(조립 및 링크) 시 지정되며 프로그램 실행 중에 변경할 수 없다.
- Heap- (주소 0x1004 0000 - 스택 데이터에 도달할 때까지 위로 커짐) 런타임에 필요에 따라 할당되는 동적 데이터다. 이 메모리가 할당되고 회수되는 방식은 언어별로 다르다. 힙의 데이터는 항상 전역적으로 사용할 수 있다.
- Stack- (주소 0x7fff fe00 - 힙 데이터에 도달할 때까지 아래로 증가) 프로그램 스택은 push 및 pop 작업을 통해 서브프로그램에 할당된 동적 데이터다. 모든 메소드 로컬 변수가 여기에 저장된다. push 및 pop 작업의 특성으로 인해 생성할 스택 레코드의 크기는 프로그램이 어셈블될 때 알려야 한다.
- Kernel- (주소 0x9000 0000 - 0xffff 0000) - 운영 체제에서 커널 메모리를 사용하므로 사용자가 액세스할 수 없다.
- MMIO - (주소 0xffff 0000 - 0xffff 0010) - 메모리 매핑된 I/O 로 모니터, 디스크 드라이브, 콘솔 등과 같이 메모리에 없는 모든 유형의 외부 데이터에 사용된다.

3. 시뮬레이션 과정, 결과 분석

① Hello World.asm을 사용해 학번과 영문 이름을 출력하고 설명한다.

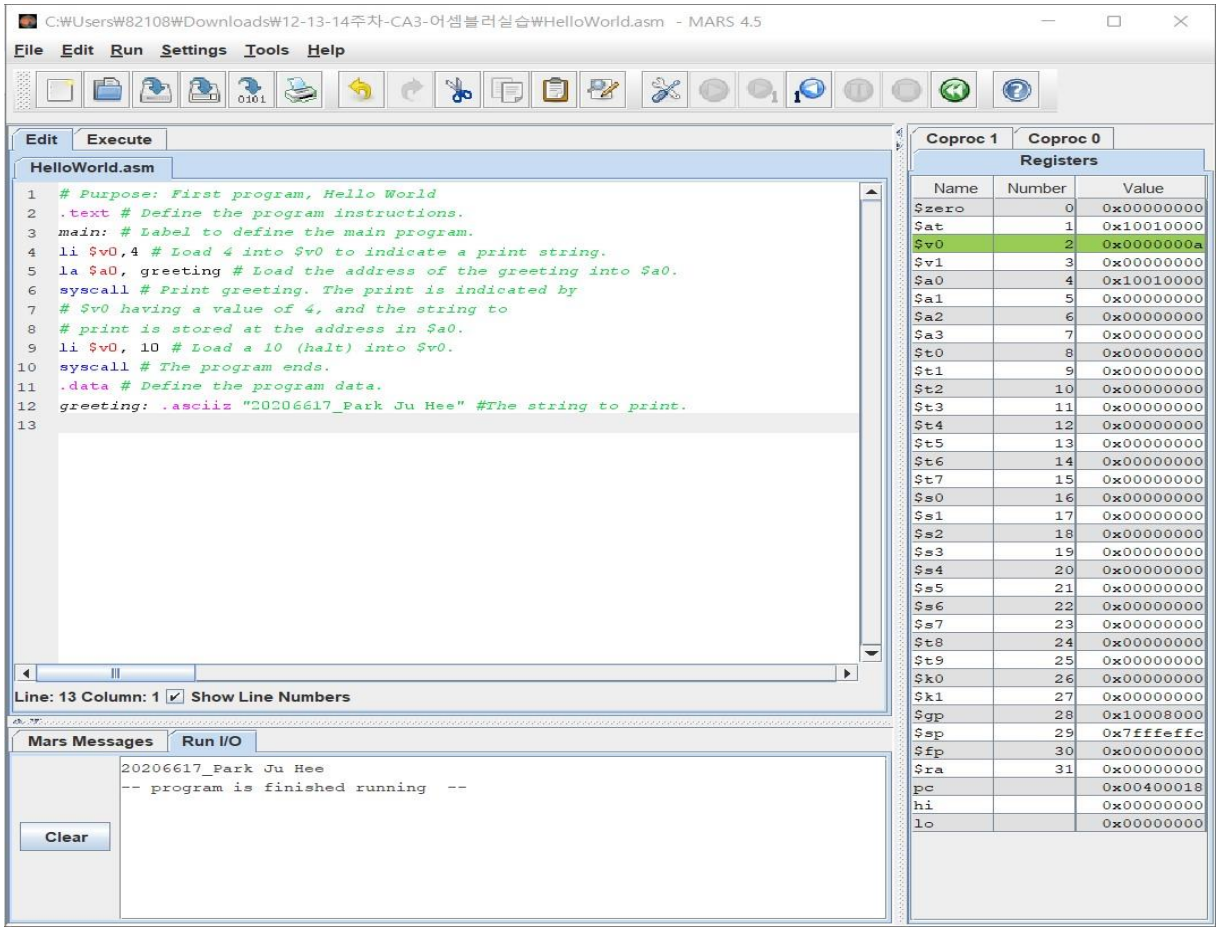
명령어는 연산자와 인수다. 따라서 li은 연산자이고 li $v0,4는 명령이다. 상수는 명령어 자체에 있어야 하므로 명령어 li $a0, 4의 값 4는 즉치값이지만 문자열 "20206617_Park Ju hee"는 상수이지만 즉각적인 값은 아니다. 텍스트 세그먼트에는 지침과 레이블만 정의할 수 있고 데이터 세그먼트에는 데이터와 레이블만 정의할 수 있다. 연산자는 li, la 및 syscall과 같은 텍스트 문자열이다.

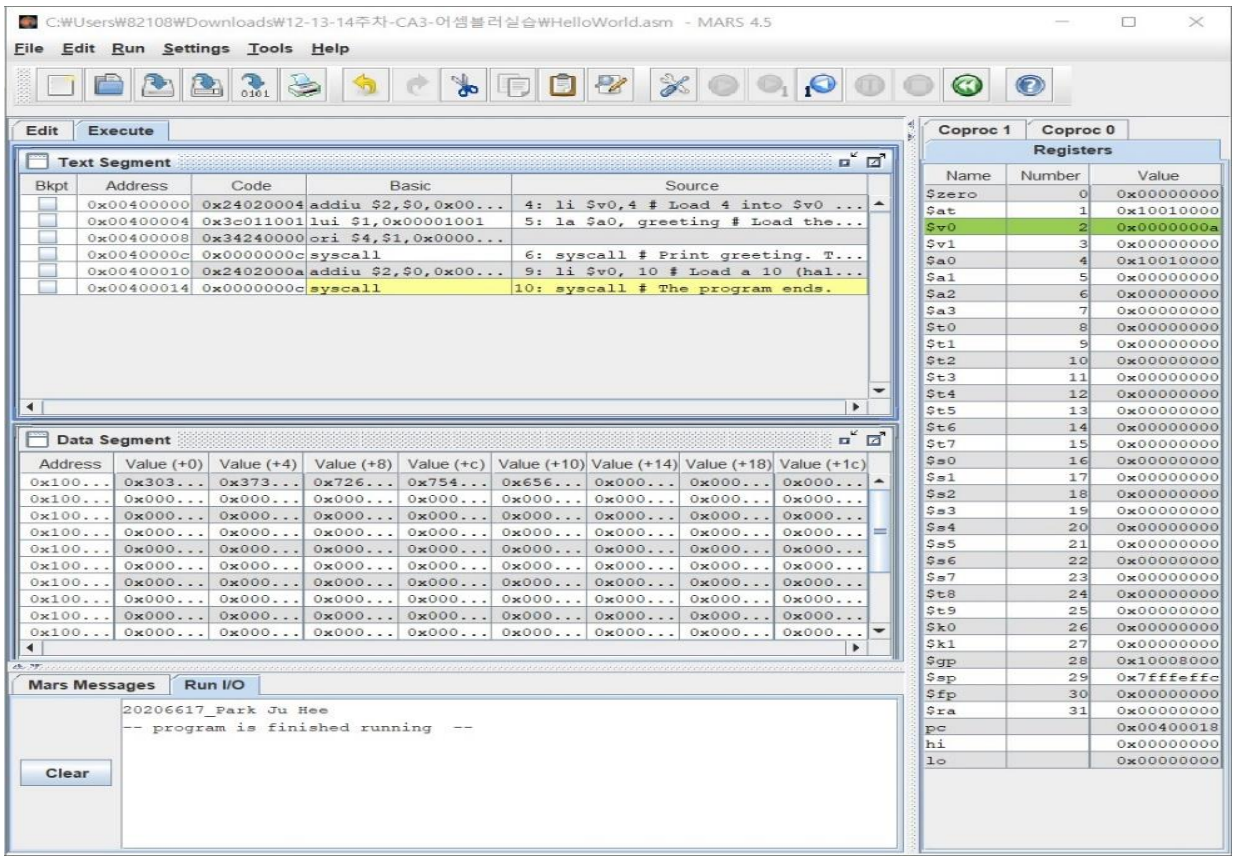
- li는 즉치값을 레지스터에 로드 하는 것을 의미한다. (li $v0, 4는 $v0 <- 4)

- la는 레이블의 주소를 레지스터에 로드 하는 것을 의미한다. (la $a0, greeting 은 $a0 <- greeting)

- syscall은 시스템 서비스를 요청하는 데 사용된다. 실행할 서비스는 $v0 레지스터에 포함된 숫자이다. 서비스 4는 $a0에 포함된 메모리 주소에서 시작하는 문자열 greeting을 print해준다. 서비스 10은 프로그램을 중단하거나 종료한다.

- .asciiz 지시문은 어셈블러에게 ASCII 문자열로 뒤따르는 데이터를 해석하도록 지시한다. 마지막 줄 greeting에 문자열 "20206617_Park Ju hee"을 저장해 MARS 편집 창에서 모든 MIPS 어셈블러 지시문을 제공한다.


C:₩Users₩82108₩Downloads₩12-13-14주차-CA3-어셈블러실습₩HelloWorld.asm - MARS 4.5
File Edit Run Settings Tools Help
Edit Execute
HelloWorld.asm
1 # Purpose: First program, Hello World
2 .text # Define the program instructions.
3 main: # Label to define the main program.
4 li $v0,4 # Load 4 into $v0 to indicate a print string.
5 la $a0, greeting # Load the address of the greeting into $a0.
6 syscall # Print greeting. The print is indicated by
7 # $v0 having a value of 4, and the string to
8 # print is stored at the address in $a0.
9 li $v0, 10 # Load a 10 (halt) into $v0.
10 syscall # The program ends.
11 .data # Define the program data.
12 greeting: .asciiz "20206617_Park Ju Hee" #The string to print.
13
Line: 13 Column: 1 Show Line Numbers
Mars Messages Run I/O
20206617_Park Ju Hee
-- program is finished running --
Clear
Coproc 1 Coproc 0
Registers
Name Number Value
$zero 0 0x00000000
$at 1 0x10010000
$v0 2 0x0000000a
$v1 3 0x00000000
$a0 4 0x10010000
$a1 5 0x00000000
$a2 6 0x00000000
$a3 7 0x00000000
$t0 8 0x00000000
$t1 9 0x00000000
$t2 10 0x00000000
$t3 11 0x00000000
$t4 12 0x00000000
$t5 13 0x00000000
$t6 14 0x00000000
$t7 15 0x00000000
$s0 16 0x00000000
$s1 17 0x00000000
$s2 18 0x00000000
$s3 19 0x00000000
$s4 20 0x00000000
$s5 21 0x00000000
$s6 22 0x00000000
$s7 23 0x00000000
$t8 24 0x00000000
$t9 25 0x00000000
$k0 26 0x00000000
$k1 27 0x00000000
$gp 28 0x10008000
$sp 29 0x7fffeffc
$fp 30 0x00000000
$ra 31 0x00000000
pc 0x00400018
hi 0x00000000
lo 0x00000000


C:₩Users₩82108₩Downloads₩12-13-14주차-CA3-어셈블러실습₩HelloWorld.asm - MARS 4.5
File Edit Run Settings Tools Help
Edit Execute
Text Segment
Bkpt Address Code Basic Source
0x00400000 0x24020004 addiu $2,$0,0x00... 4: li $v0,4 # Load 4 into $v0 ...
0x00400004 0x3c011001 lui $1,0x00001001 5: la $a0, greeting # Load the...
0x00400008 0x34240000 ori $4,$1,0x0000...
0x0040000c 0x0000000c syscall 6: syscall # Print greeting. T...
0x00400010 0x2402000a addiu $2,$0,0x00... 9: li $v0, 10 # Load a 10 (hal...
0x00400014 0x0000000c syscall 10: syscall # The program ends.
Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+c) Value (+10) Value (+14) Value (+18) Value (+1c)
0x100... 0x303... 0x373... 0x726... 0x754... 0x656... 0x000... 0x000... 0x000...
Mars Messages Run I/O
20206617_Park Ju Hee
-- program is finished running --
Clear
Coproc 1 Coproc 0
Registers
Name Number Value
$zero 0 0x00000000
$at 1 0x10010000
$v0 2 0x0000000a
$v1 3 0x00000000
$a0 4 0x10010000
$a1 5 0x00000000
$a2 6 0x00000000
$a3 7 0x00000000
$t0 8 0x00000000
$t1 9 0x00000000
$t2 10 0x00000000
$t3 11 0x00000000
$t4 12 0x00000000
$t5 13 0x00000000
$t6 14 0x00000000
$t7 15 0x00000000
$s0 16 0x00000000
$s1 17 0x00000000
$s2 18 0x00000000
$s3 19 0x00000000
$s4 20 0x00000000
$s5 21 0x00000000
$s6 22 0x00000000
$s7 23 0x00000000
$t8 24 0x00000000
$t9 25 0x00000000
$k0 26 0x00000000
$k1 27 0x00000000
$gp 28 0x10008000
$sp 29 0x7fffeffc
$fp 30 0x00000000
$ra 31 0x00000000
pc 0x00400018
hi 0x00000000
lo 0x00000000

② Addition.asm을 사용해 학번을 두자리씩 더해 화면에 출력하기

5행: li $t1, 20는 $ t1<- 20

6행: li $t2, 20는 $ t2<- 20

7행: add 연산자를 사용해 정수를 포함하는 $t1 및 $t2 레지스터의 값인 20을 가져와 더한 다음 값을 다시 $t0 레지스터에 저장한다. $t0에 40이 저장된다. (20+20=40)

9행: addi 연산자는 $t0의 값(40)을 가져와 40에 66을 더하고 결과(106)를 다시 $t0에 저장한다.

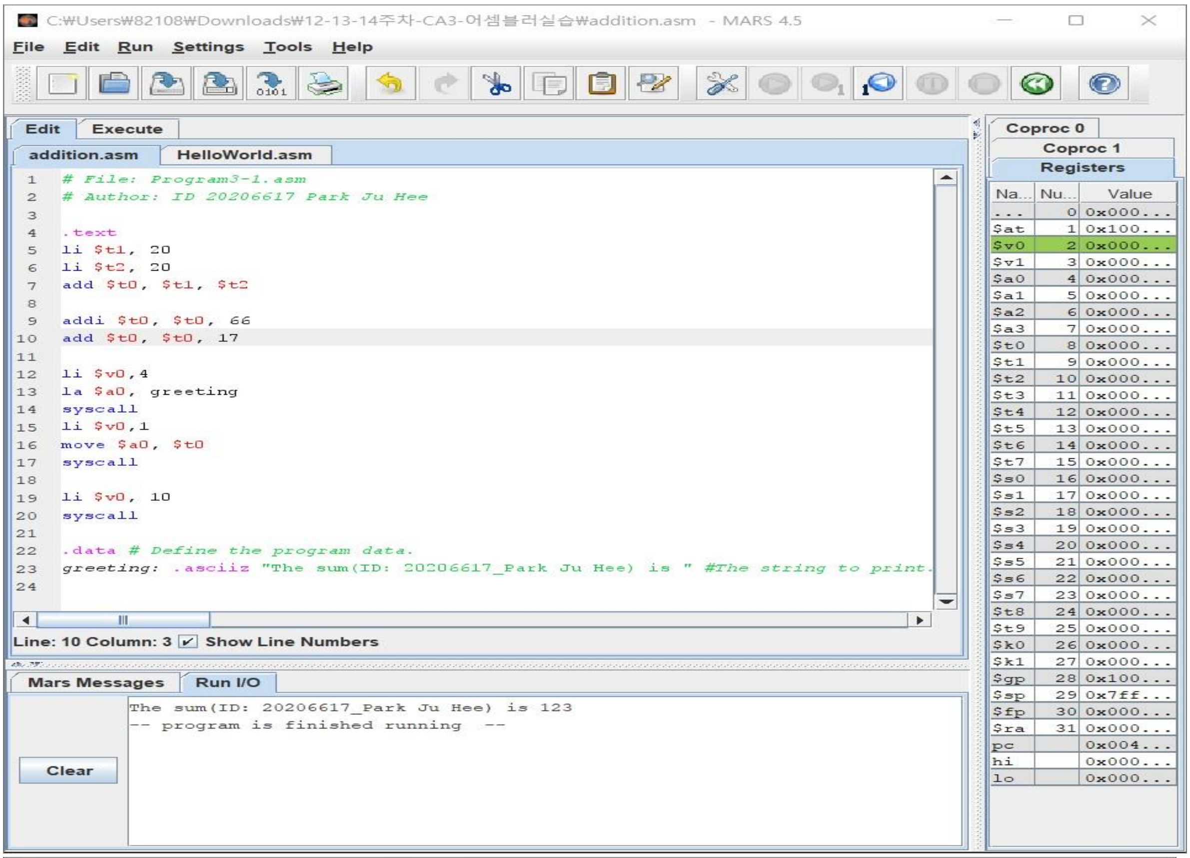
10행: add연산자를 사용해 $t0(106)의 값을 가져와 17을 더한 결과(123)를 다시 $t0에 저장한다.

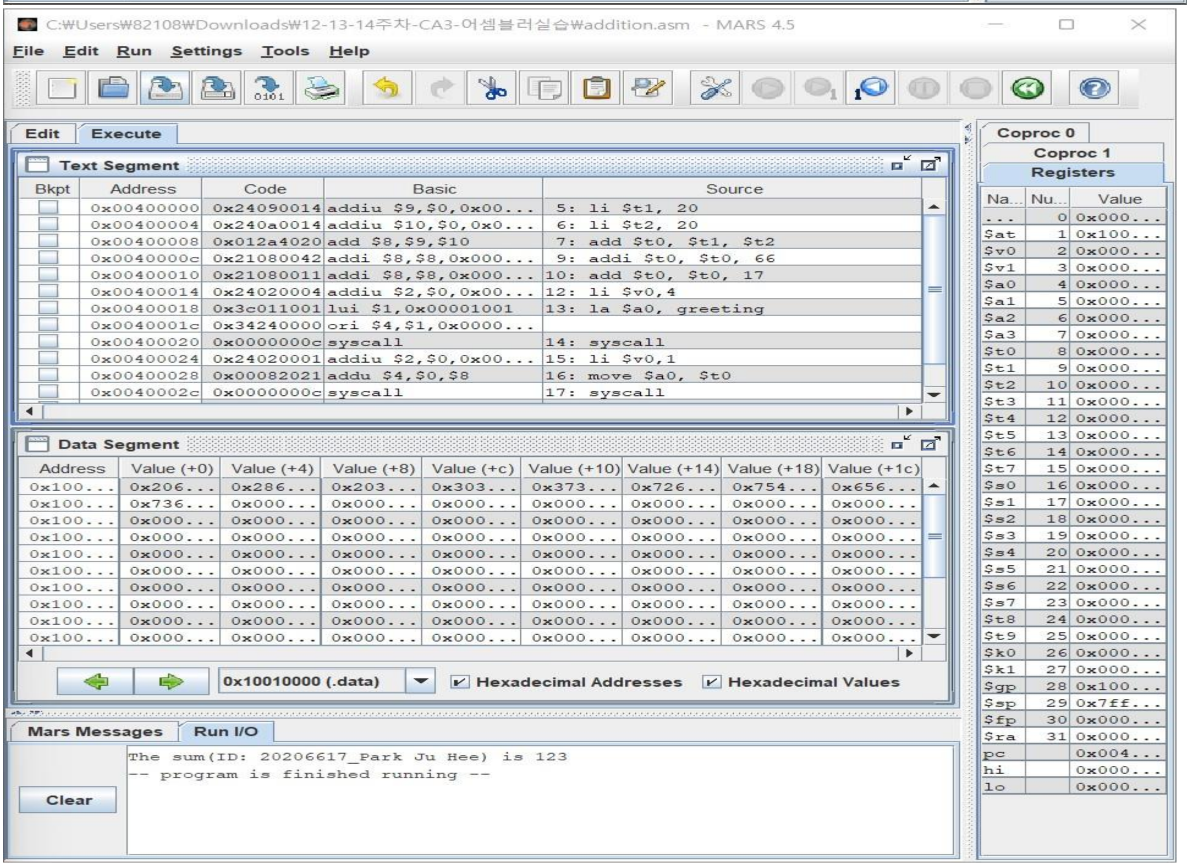
12행: 실행할 서비스는 $v0 레지스터에 포함된 숫자다. 서비스 4는 $a0에 포함된 메모리 주소에서 시작하는 문자열 greeting을 print해준다.

15행: 서비스 1은 $t0의 값을 $a0위치로 옮겨 greeting 다음에 $t0가 출력된다.

19행: 서비스 10은 프로그램을 중단한다.

23행: .asciiz 지시문은 어셈블러에게 ASCII 문자열로 뒤따르는 데이터를 해석하도록 지시한다. 마지막 줄 greeting에 문자열 "The sum (Id: 20206617_Park Ju hee) is"을 저장해 MARS 편집 창에서 모든 MIPS 어셈블러 지시문을 제공한다.


C:₩Users₩82108₩Downloads₩12-13-14주차-CA3-어셈블러실습₩addition.asm - MARS 4.5
File Edit Run Settings Tools Help
Edit Execute
addition.asm HelloWorld.asm
File: Program3-1.asm
Author: ID 20206617 Park Ju Hee
.text
li $t1, 20
li $t2, 20
add $t0, $t1, $t2
addi $t0, $t0, 66
add $t0, $t0, 17
li $v0,4
la $a0, greeting
syscall
li $v0,1
move $a0, $t0
syscall
li $v0, 10
syscall
.data # Define the program data.
greeting: .asciiz "The sum(ID: 20206617_Park Ju Hee) is " #The string to print.
Line: 10 Column: 3 Show Line Numbers
Mars Messages Run I/O
The sum(ID: 20206617_Park Ju Hee) is 123
-- program is finished running --
Clear
Coproc 0 Coproc 1 Registers


C:₩Users₩82108₩Downloads₩12-13-14주차-CA3-어셈블러실습₩addition.asm - MARS 4.5
File Edit Run Settings Tools Help
Edit Execute
Text Segment
Bkpt Address Code Basic Source
0x00400000 0x24090014 addiu $9,$0,0x00... 5: li $t1, 20
0x00400004 0x240a0014 addiu $10,$0,0x0... 6: li $t2, 20
0x00400008 0x012a4020 add $8,$9,$10 7: add $t0, $t1, $t2
0x0040000c 0x21080042 addi $8,$8,0x000... 9: addi $t0, $t0, 66
0x00400010 0x21080011 addi $8,$8,0x000... 10: add $t0, $t0, 17
0x00400014 0x24020004 addiu $2,$0,0x00... 12: li $v0,4
0x00400018 0x3c011001 lui $1,0x00001001 13: la $a0, greeting
0x0040001c 0x34240000 ori $4,$1,0x0000...
0x00400020 0x0000000c syscall 14: syscall
0x00400024 0x24020001 addiu $2,$0,0x00... 15: li $v0,1
0x00400028 0x00082021 addu $4,$0,$8 16: move $a0, $t0
0x0040002c 0x0000000c syscall 17: syscall
Data Segment
Address Value (+0) Value (+4) Value (+8) Value (+c) Value (+10) Value (+14) Value (+18) Value (+1c)
0x10010000 (.data) Hexadecimal Addresses Hexadecimal Values
Mars Messages Run I/O
The sum(ID: 20206617_Park Ju Hee) is 123
-- program is finished running --
Clear
Coproc 0 Coproc 1 Registers