2차 실험 보고서

실습명: 동기식순서 논리 회로시뮬레이션/

어셈블러 샘플 분석

과목명: 컴퓨터 구조 2반

실습 날짜: (동기식순서 논리 회로)2021.11.10/

(어셈블리언어)2021.12.01

보고서제출일:

2021.12.01

학과/학번: 콘텐츠IT, 20185309

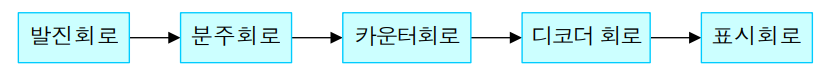
성명: 황명원

1. 동기식 순서논리 회로 (디지털 시계)
2. 실습목적

디지털 시계를 모의실험으로 구현하여 동작을 분석하여, 동기식 순서 논리 회로를 이해한다. 그리고 카운터의 주요 응용으로서 디지털 시계와 주파수 카운터의 동작 원리를 이해할 수 있다.

1. 기본이론

구성할 회로의 블록 다이어그램을 살펴보면 다음과 같다.



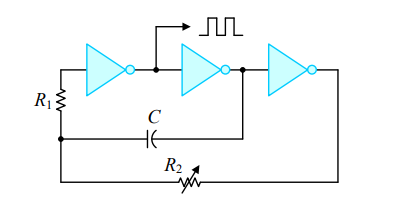
1)발진회로

디지털 시계에 안정적인 클록(clock)을 제공하는 회로이며

가변저항 R2 (100KΩ)를 가변하여 발진주파수를 변화시킨다.

또 가변저항을 적절히 변화시켜서 디지털 시계의 시간을 조정해주는 역할을 한다.

발전회로의 기본적인 회로는 다음과 같다.

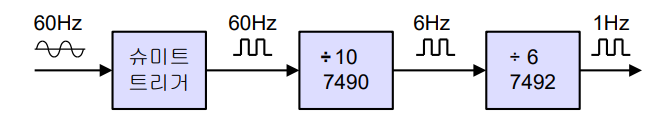


2)분주회로

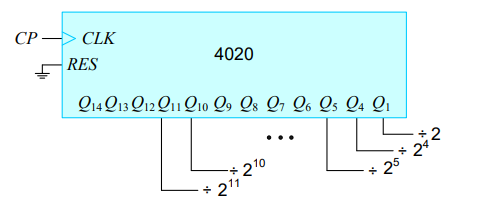
발진회로로부터 얻어진 구형파를 이용하여 디지털 시계의 기본단위인 1초를 나타내기 위한 1Hz 주파수를 얻는 회로이다. 시계를 만드는데 있어서 시간은 클락의 변함에 따라 만들어야 하는데 이 회로를 통해 주기적으로 전압이나 전류가 변하는 신호를 만들어준다.

(출력파형은 정현파, 구현파 ,삼각파, 왜율 이 있고 그중에 구현파를 이용한다.)

구형파를 얻는 예시 이미지는 다음과 같다.

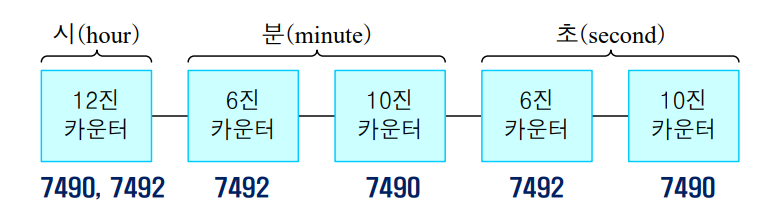


그렇게 4020을 이용하여 1Hz 구형파를 얻는 회로는 다음과 같다.



3)카운터 회로

우선 시, 분, 초로 나눈 기본적인 회로 구성은 다음과 같다.



분과 초는 회로상 똑같은 구조로 6진카운터-10진카운터이고 시(hour)만 12진 카운터이다.

각 카운터에대해 보면

ㅡ7492: 2진 카운터(mod-2)와 6진 카운터(mod-6)가 독립적으로 내장된다 (참고: 사용법은 7493에 준한다.)

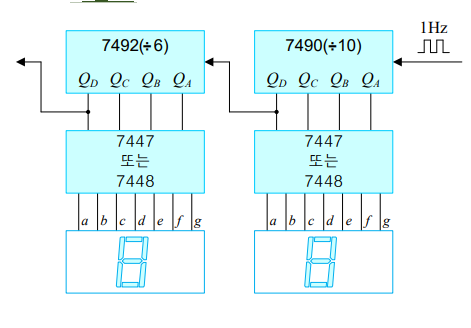
ㅡ7490: 2진 카운터(mod-2)와 5진 카운터(mod-6)가 독립적으로 내장된다 (참고: 사용법은 7493에 준한다.)

신호가 카운터로 들어오면 카운터와 세그먼트가 연결되있는데 카운트 될때마다 연결되있는 세그먼트에 숫자가 하나씩 올라가게된다.

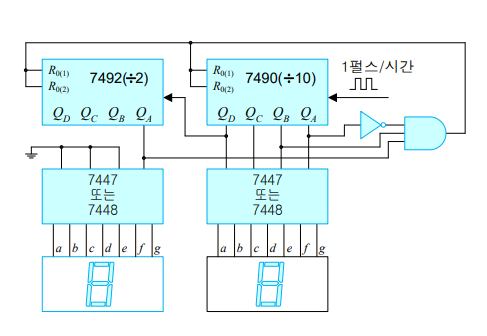
4)디코더 회로 및 표시회로

카운터 회로의 출력인 2진 데이터를 표시하기 위해 디코더 회로 및 표시회로가 필요하다. 이번 실습에서 디코더 회로는 7447을 이용하고 표시회로는 7-segment로 시간을 나타낸다.

분, 초 단위의 카운터, 디코더 및 드라이브 회로는 다음과 같다.



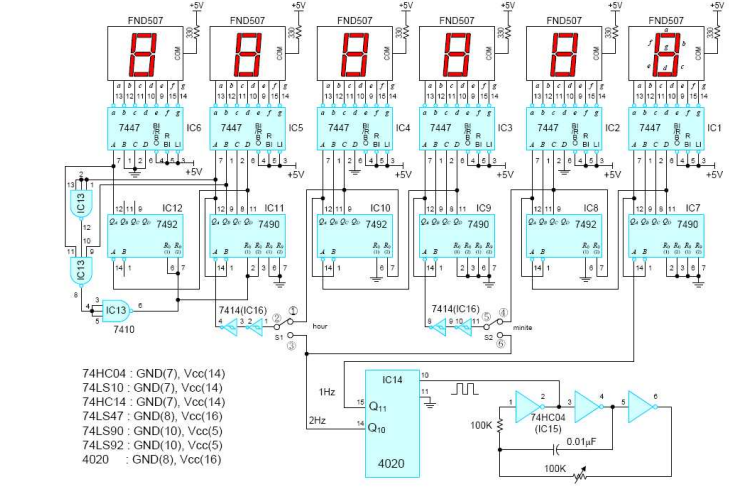
시 단위의 카운터, 디코더 및 드라이브 회로는 다음과 같다.



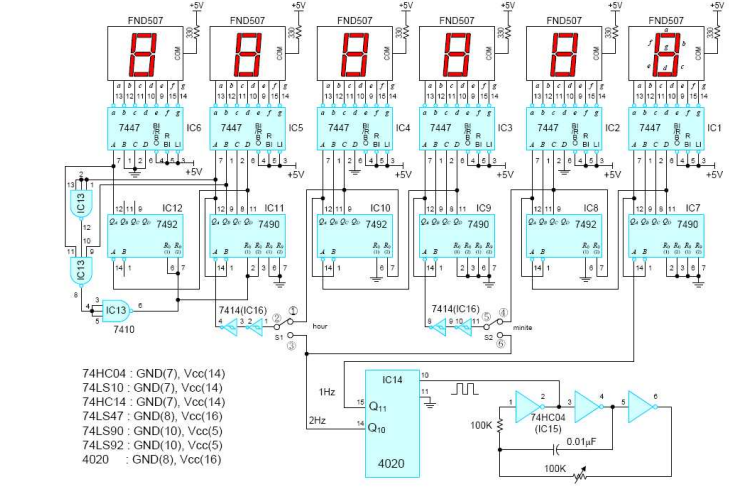
위 사진에 보이는 7492와 7490 덕분에 1초를 나타내는 1Hz를 구할수 있다.

(기타사항: 시와 분을 조정하는 S1과 S2 스위치 부분에 사용한 7414는 슈미트 트리거(Schmitt trigger) 회로가 내장된 IC로서 기계적인 스위치 조작 시에 발생하는 접점(contact)의 바운싱(bouncing) 영향을 제거하기 위해 사용한다.)

이렇게 공부한 이론적 지식으로 만들어진 디지털 시계 회로는 사진과 같다.

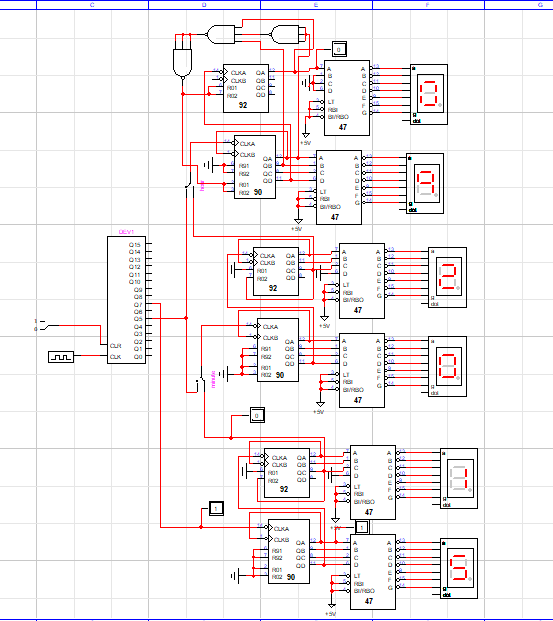


1. 시뮬레이션 과정



위 사진을 참고하여 회로를 작성할려고 했는데 몇가지 다른 부분이 있

었다. 첫번째로 저항이나 커패시터 등과 같은 수동소자는 사용하지 않아도 된다. 따라서 발진회로는 회로도와 같이 구성할 필요가 없으므로 “clock”만을 사용한다. 7-세그먼트의 COM 단자에 연결한 330Ω 저항도 시뮬레이션 수행에는 필요하지 않다. 7414는 실제적인 스위치 조작 시에 발생하는 문제점을 해결하기 위해 사용한 IC이므로 사 용할 필요가 또한 없다. 7-세그먼트는 라이브러리에서 “7-Seg Disp Inv”를 사용하고, 시와 분을 조정하는 스위치 S1과 S2는 “SPDT Switch”를 라이브러리에서 선택하여 사용한다. 이러한 다른 점을 알고서 회로를 만들어보았다.



클럭은 계속해서 전달 될 것이고 2진,6진,10진 카운터의 동작하는 내용을 간략하게 설명하자면 다음과 같다.

초의 일의자리 10진 카운터 증가시점 : 1 Hz 클럭이 인가될 때마다

초의 십의자리  6진 카운터 증가시점 : 초의 일의자리 10진 카운터 가 9에서 0으로 변할 때

분의 일의자리 10진 카운터 증가시점 : 초의 십의자리  6진 카운터 가 5에서 0으로 변할 때 (즉, 59초에서 00초로 변할 때)

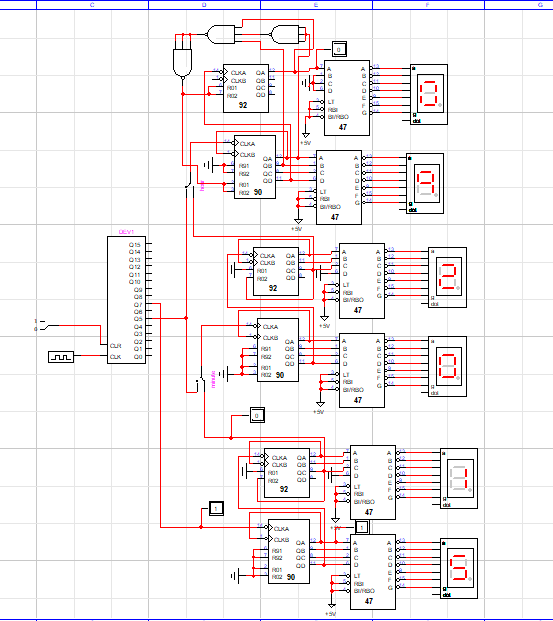
분의 십의자리  6진 카운터 증가시점 : 분의 일의자리 10진 카운터 가 9에서 0으로 변할 때 (즉, x9분59초에서 y9분 00초로 변할 때)

시를 나타내는 12진 카운터 증가시점 : 분의 십의자리  6진 카운터 가 5에서 0으로 변할 때(즉, 59분59초에서 00분00초로 변할 때)

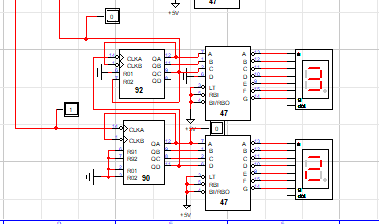
위 과정들로 하여 초,분,시 들이 진짜 시간처럼 1씩 증가하여 각 일의 자리와 십의 자리들의 숫자가 바뀌었다.

1. 실험결과와 분석

원래 사진으로 나와있던 회로에서 약간의 수정사항(위에서 몇가지 다른부분이라고 언급했던 부분)들이 있어서 그것을 바꾸고 회로를 작성하여 실험을 해보았다. 처음에는 회로를 다 작성해도 잘안됬었지만 교수님께서 알려주신대로 Reset Simulation 과 Clear Unknowns 를 누르고 실행해보니 정말로 1초마다 시간이 올라가는 것처럼 7-세그먼트의 숫자가 하나씩 올라갔다.

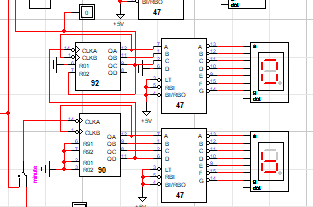


초 ㅡ 실험결과



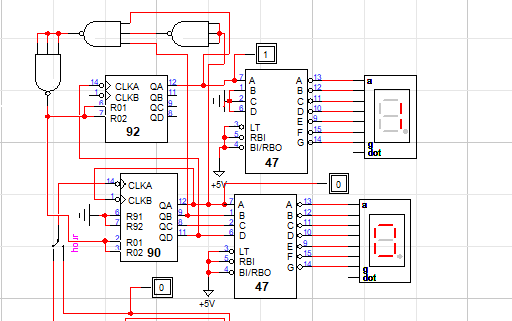
사진으로 올려서 보이진 않지만 클럭이 들어오고 들어감에 따라 밑에 있는 두개의 바이너리프로브가 주기적으로 0과 1로 바뀌는 모습을 봤다. 들어옴과 동시에 초의 일의 자리 부분인 숫자가 계속 바뀌는 모습도 확인했다.

분 ㅡ 실험결과



분 의 회로는 초의 회로와 동일하다. 다만 다른점은 초에서는 처음 생성되는 클락의 신호로 세그먼트가 표시되어 1초씩 올라가는걸로 설정이 되었지만 분 에서는 초의 6진 카운터가 다시 0이될 때 분의 10진카운터에 신호를 주어 카운트를 증가시켜줬다. 그래서 초가 59초에서 00초가 되는순간 분은 00분 에서 01분이 되는걸 확인할 수 있었다.

시간 ㅡ 실험결과



시간도 신호가 잘 전달되는가 확인하기위해 바이너리프로브를 연결하여 확인해봤고 분에서 59분에서 00분이 될 때 시간은 00시간에서 01시간으로 바뀐걸 확인했다.

디지털시계를 만듬으로써 회로를 이용해 카운트를 누적시키는 방법을 알았고 누적된 카운트로 카운트를 세는것도 구현해서 신기했다.

# 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.

1. 기본이론

1)어셈블리 언어 특징

-GUI기반 통합개발 환경이다.

-스프레드시트 모양의 레지스터이고 ,메모리 값 표시 및 수정한다.

-다양한 속도의 단일 스텝으로 실행한다.

-MARS 실행 데이터에 연동한 디바이스 시뮬레이션 툴을 제공한다.

(MIPS X-Ray, 명령어 통계, MIPS 데이터패스, 캐시, BHT 등)

2)어셈블리 지시어

-기계어로 변환되지 않고 어셈블러에게 정보를 전달하는 지시어 이다.(“.”으로 시작)

3)주요 지시어

-섹션(세그먼트) 시작: .text .data

-전역 심볼 선언: .globl

-데이터 표시: .byte .half .word .ascii .asciiz .float .double

-워드 경계로 정렬: .align

1. 시뮬레이션 과정, 결과 분석
2. Hello World.asm을 사용해 학번과 영문 이름을 출력하고 설명한다.

(코드 와 실행결과)

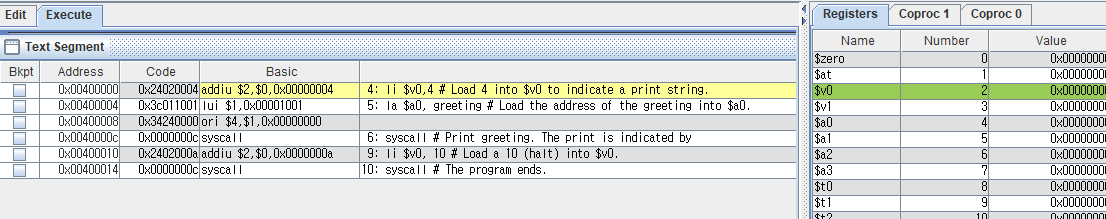
텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

(실행결과 순서대로 설명)

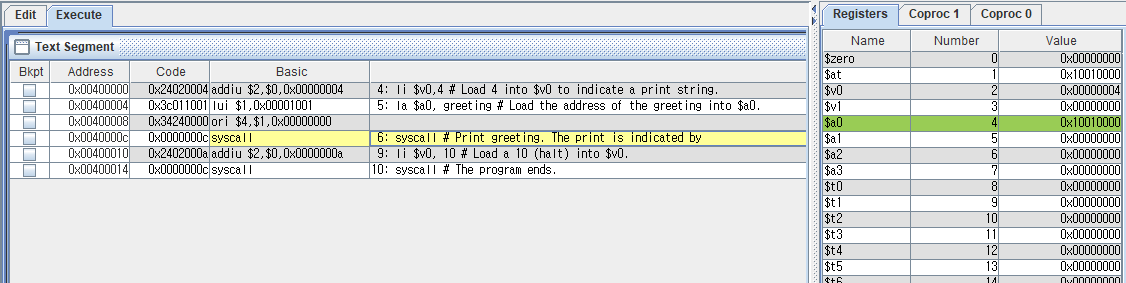


4를 v0 레지스터에 저장한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

la 는 a0 라는 레지스터에 greeting 의 값을 로드하도록 해준다.



로드가 끝났으면 출력을 해볼건데 출력을 하기위해 syscall(System call code)를 쓴다.

테이블이(가) 표시된 사진

자동 생성된 설명

테이블이(가) 표시된 사진

자동 생성된 설명

li $v0, 10

syscall

이 코드로 코드가 끝났다는 것을 알려준다.

1. Addition.asm을 사용해 학번 (예 20211234)에서 두자리 씩 더해 화면에 출력 하기 20+21+12+34=87를 화면 출력하고 설명한다.

(코드 와 실행결과)

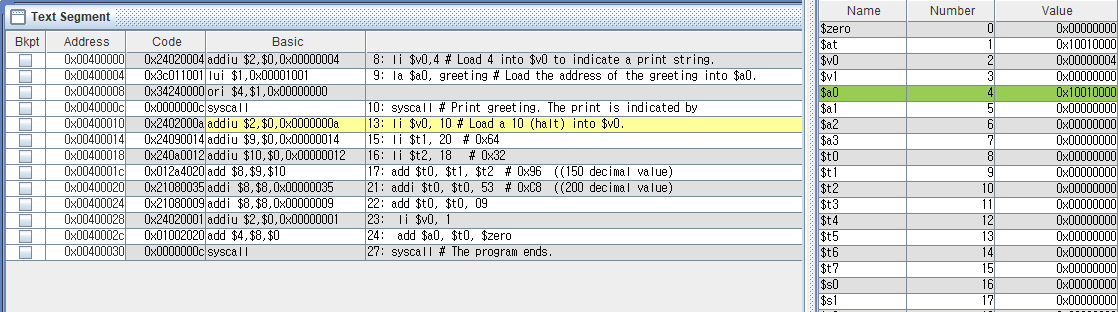
텍스트이(가) 표시된 사진

자동 생성된 설명

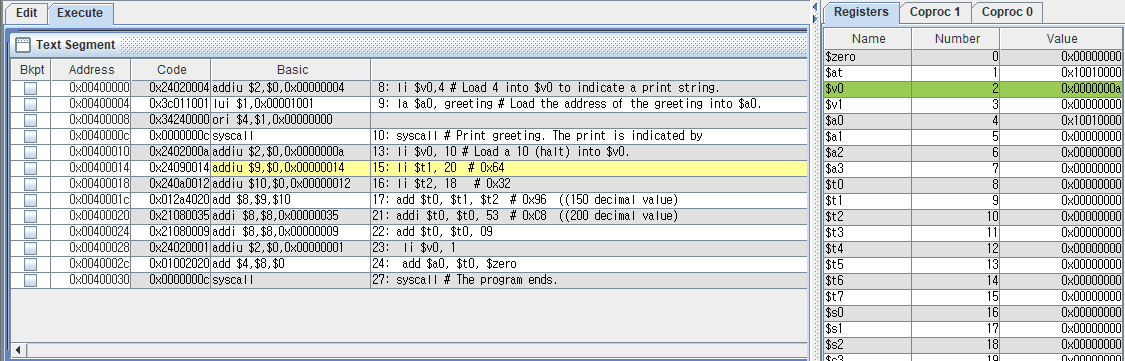
텍스트이(가) 표시된 사진

자동 생성된 설명

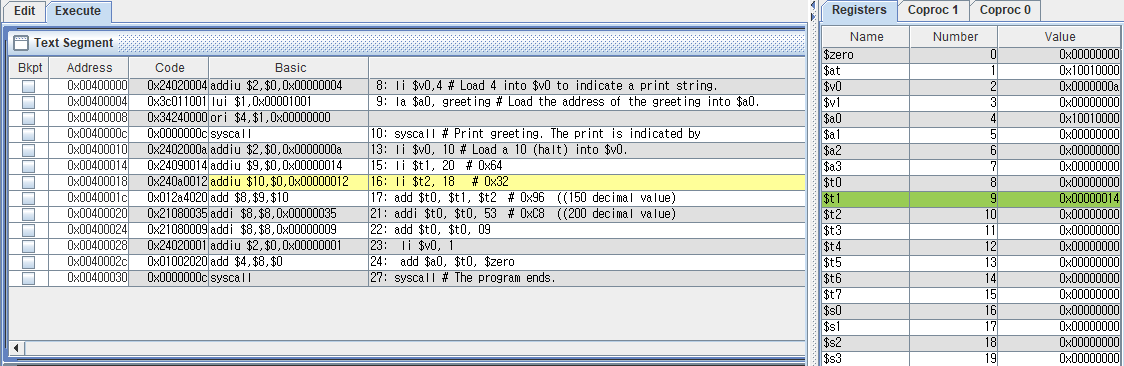
(실행결과 순서대로 설명)



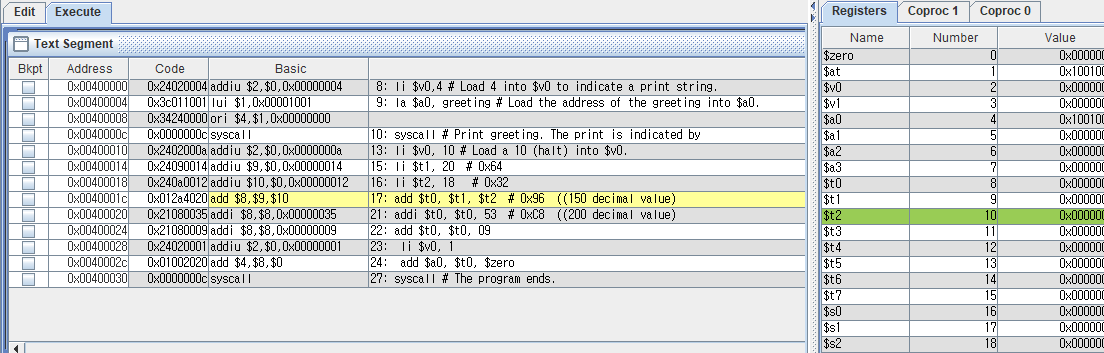
(\* 이름출력하는것은 위와 같은 방법이므로 여기선 다시 적지 않았습니다.)



t1이라는 레지스터에 20을 저장한다. (t0~t7 레지스터는 값을 임시로 저장하는 레지스터이다.)



t2 라는 레지스터에 18을 저장한다.

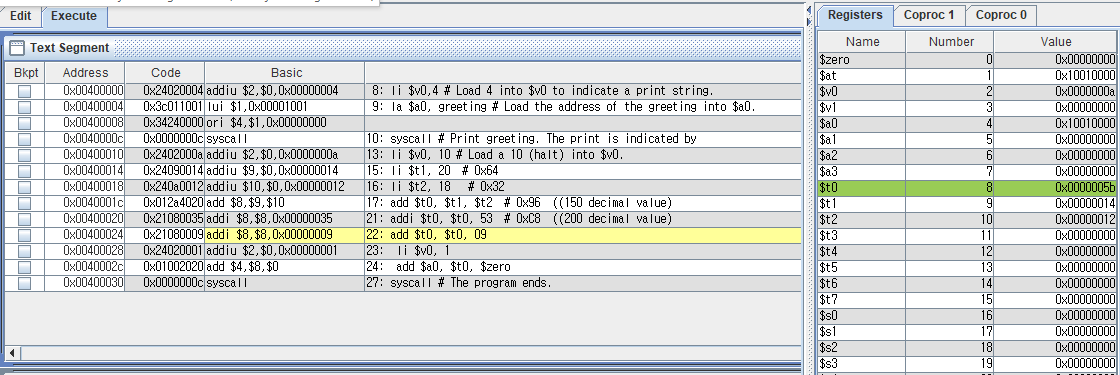


add를 사용해 더하기를 수행한다. t1과 t2를 더한값을 t0에 저장한다.

테이블이(가) 표시된 사진

자동 생성된 설명

addi는 상수를 더할 때 쓰는 명령어로 t0와 53을 더한값을 t0에 저장한다.



마찬가지로 t0와 09를 더해 t0에 저장한다.

테이블이(가) 표시된 사진

자동 생성된 설명

t0와 0을 더해 a0에 저장한다. (a0는 함수 인수 저장용 레지스터이다.)

텍스트이(가) 표시된 사진

자동 생성된 설명

프로그램의 끝을 알리고 종료시킨다.