Universidad autónoma de baja california

Ingeniero en computación

microcontroladreos

practica 2

Maestro: Jose Isabel Garcia Rocha

Erik Garcia Chávez 01275863

Lunes 2 de septiembre del 2024

Teoría previa:

el microcontrolador tiene 3 registros de 8 bits con los que administra los puertos, en el caso del 2560 tiene 11 que van del PA al PK.

estos 3 registros son:

DDRx donde x es el puerto, la letra conla que esta se identifica, donde se determina si un bit es de entrada fijandose en 0 o de salida en 1.

PORTx contrala si el pin esta en HIGH o LOW, este define o no la existenica de un resistor en pull-up interno. esto permite sacar por el puerto infromacion al mundo.=

PINx permite leer el estado de los pines de un puerto este es solo para lectura

Tabla de configuraron:

DDxn	PORTxn	PUD (in MCUCR)	I/O	Pull-up	Comment
0	0	X	Input	No	Tri-state (Hi-Z)
0	1	0	Input	Yes	Pxn will source current if ext. pulled low
0	1	1	Input	No	Tri-state (Hi-Z)
1	0	X	Output	No	Output Low (Sink)
1	1	X	Output	No	Output High (Source)

En un puerto cada pin se comporta de manera independiente, ¿cómo podemos configurar que un pin se comporte de tal manera?, para eso nos apoyaremos en la tabala, en donde depeindindo de los estados de DDRX y PORTx, el pin será de entrada o de salida, DDRx siempre esta en 0.

Esto se puede hacer en ensamblador de la siguiente manera:

ddra &=~(1<<pa0); //pa0 es entrada (pa0 indica que el pin 0 es entrada)

porta &=~(1<<pa0); //se quiere apagar el mismo bit, para que al inicio sea 0, porque port controla el pull up, pero como yta hay uno afuera, no es necesario que se active

Instrucciones:

Escriba un programa que invierte la posición de bits, intercambiando el orden de bits del más significativo al menos significativo, del valor dado por:

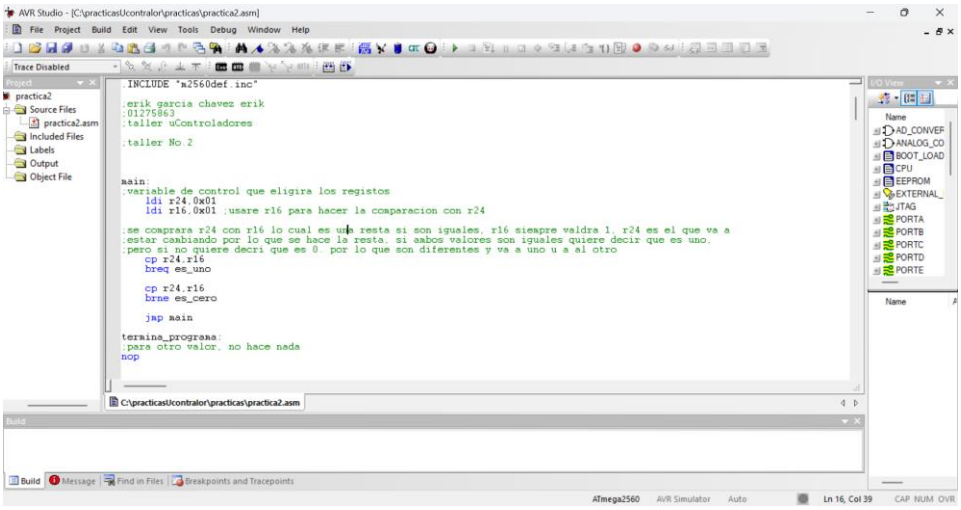
I. Los registros R31-R28 si R24 es 1
II. II. Los registros R29-R26 si R24 es 0

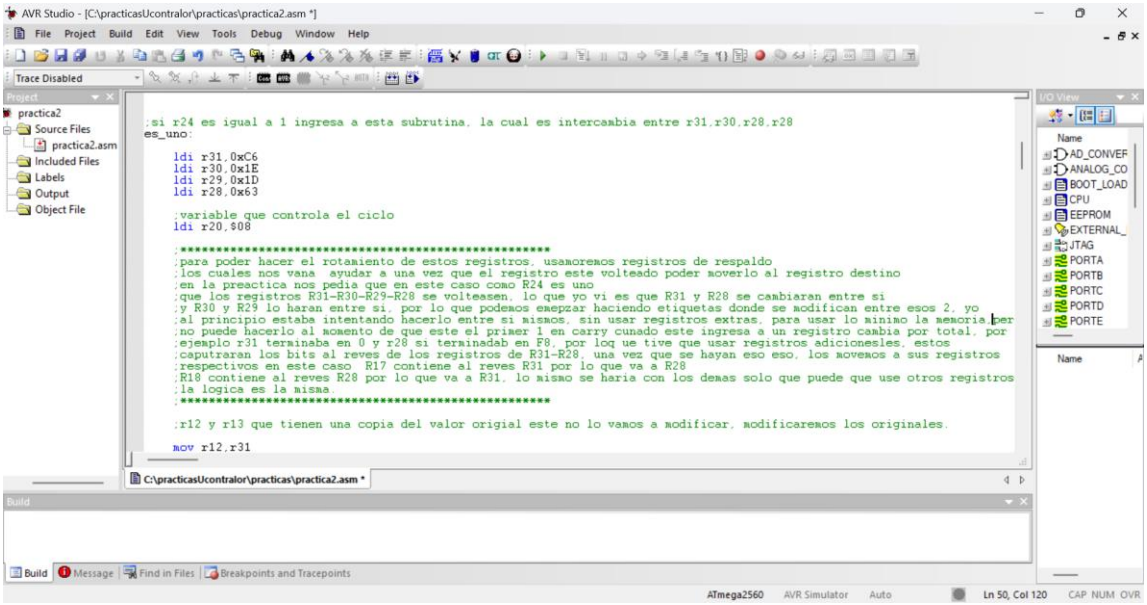
Código del programa:

Esta es mi subrutina principal, en la cual se establece el primer parámetro que se mencionó, en el cual si R24 es igual a 1 este modificara los registros R31-R30-R29-R28

Pero si R24 es igual a 0 este modificara R29-R28-R27-R26

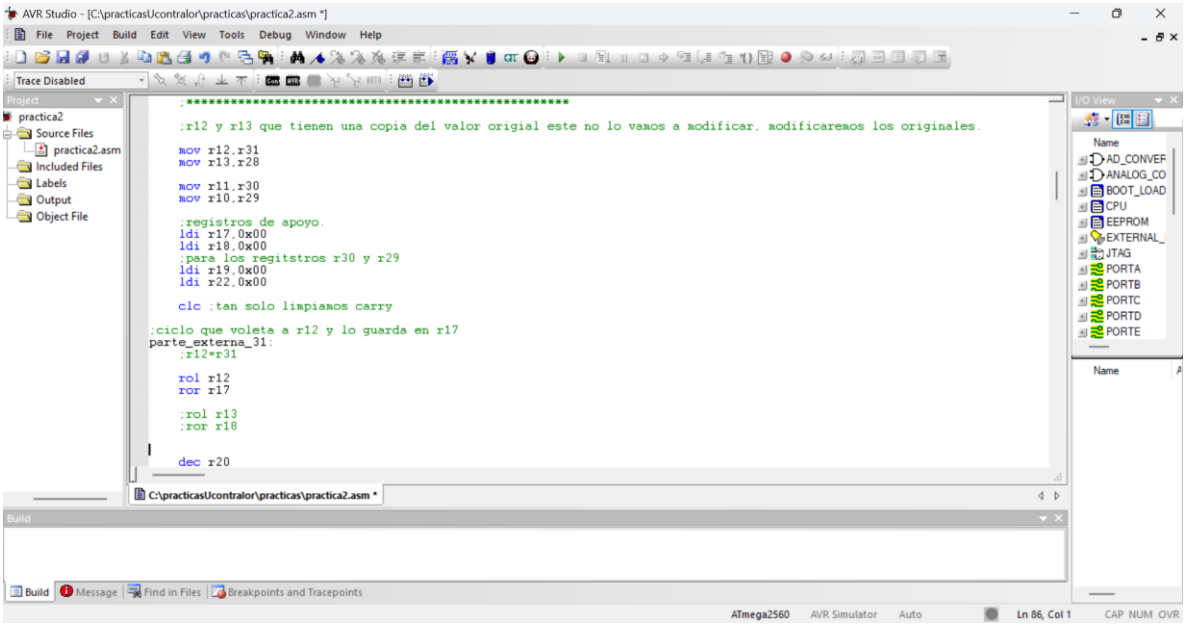
Por lo que haces una comparación en donde si son iguales quiere decir que es 1, pero si no, quiere decir que R24 es 0. Por lo que brincara depende la situación.

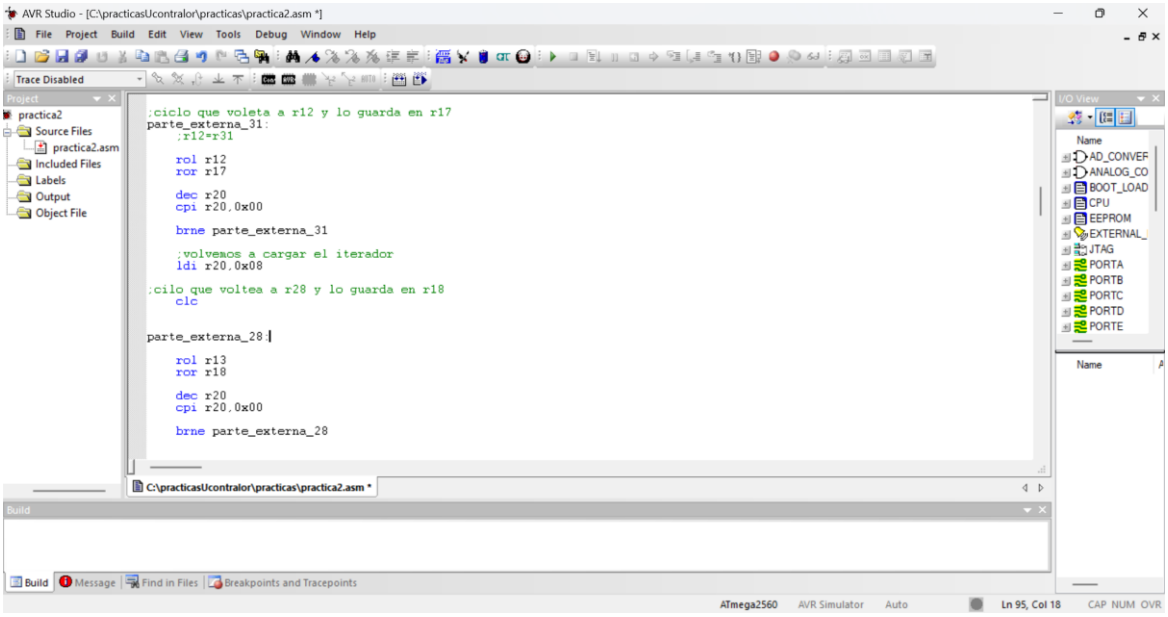


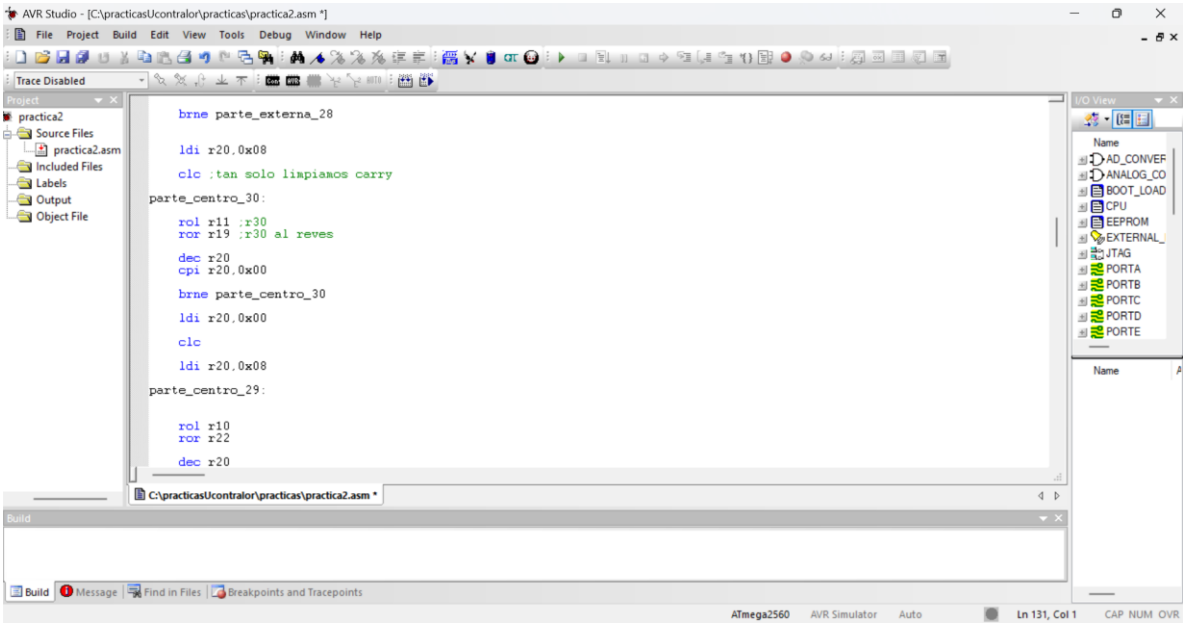


Uso 2 registros de apoyo, esto porque, en primera instancia no quiero modificar los registros originales, por lo que ocupo un grupo de 4 registros de apoyo los cuales guardaran los registros desde R31-R28, en otro grupo de 4 registros serán los que guardaran los registros, pero después de aplicar la rotación. Se hará mediante rotaciones por lo que al final los registros que tienen una copia de los registros originales tendrán un valor X, pero los otros registros que están marcado con un valor de 0H, tendrán el valor de los registros originales pero al revés, lo que se pide en la práctica, entonces con estos registros ya al revés se hace la comparación con los originales, para saber si es que si se voltea este, no debe quedar igual a como esta el registro original, por ejemplo si R31 tiene 1100 000 y R28 tiene 0000 0011, lo que es lo que hay en R31 pero al revés, no tiene caso que se intercambia, ya que quedaría igual y tan solo seria desgastar la memoria. Y se les da el valor de 0 así no afectan tanto a carry.

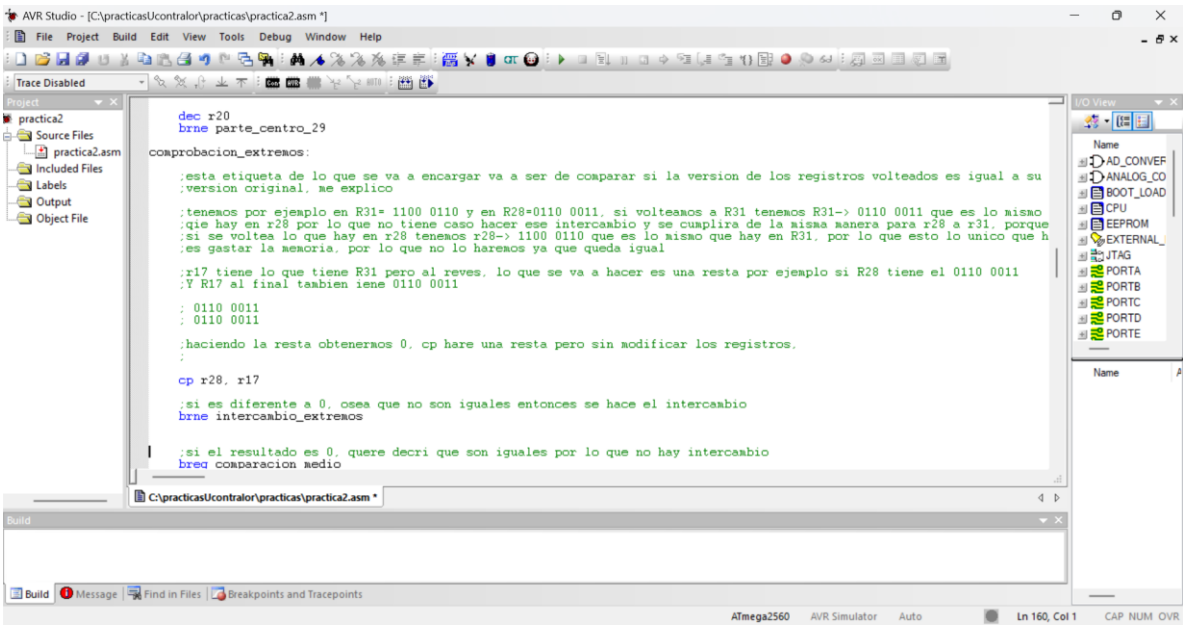
Se inicializa un contador de 8, para que cunado llegue al ultimo bit salga y no de vueltas de más, aparte de que cada registro solo es de 8 bit, y se vuelve a cargar el proceso es similar para los 4 registros.

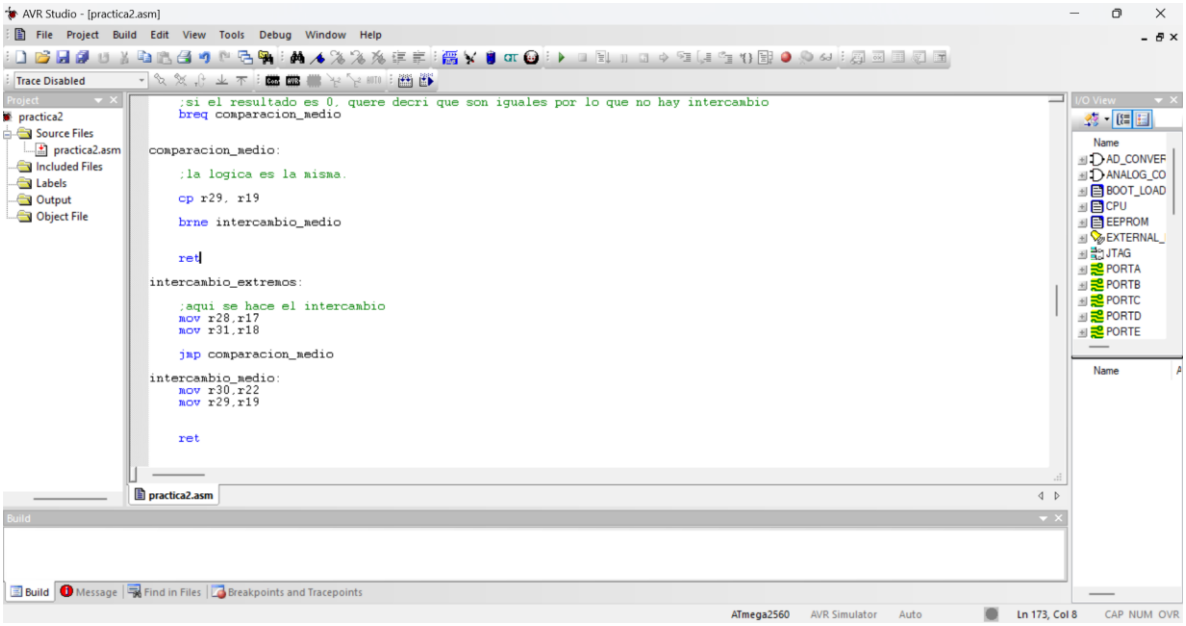

AVR Studio - [C:\practicasUcontralor\practicas\practica2.asm *]
File Project Build Edit View Tools Debug Window Help
Trace Disabled
Project
practica2
Source Files
practica2.asm
Included Files
Labels
Output
Object File
;**
;r12 y r13 que tienen una copia del valor origial este no lo vamos a modificar, modificaremos los originales.
mov r12,r31
mov r13,r28
mov r11,r30
mov r10,r29
;registros de apoyo.
ldi r17,0x00
ldi r18,0x00
;para los regitstros r30 y r29
ldi r19,0x00
ldi r22,0x00
clc ;tan solo limpiamos carry
;ciclo que voleta a r12 y lo guarda en r17
parte_externa_31:
;r12=r31
rol r12
ror r17
;rol r13
;ror r18
dec r20
C:\practicasUcontralor\practicas\practica2.asm *
Build
Build Message Find in Files Breakpoints and Tracepoints
I/O View
Name
AD_CONVEF
ANALOG_CO
BOOT_LOAD
CPU
EEPROM
EXTERNAL_
JTAG
PORTA
PORTB
PORTC
PORTD
PORTE
ATmega2560 AVR Simulator Auto Ln 86, Col 1 CAP NUM OVR


AVR Studio - [C:\practicasUcontralor\practicas\practica2.asm *]
File Project Build Edit View Tools Debug Window Help
Trace Disabled
Project
practica2
Source Files
practica2.asm
Included Files
Labels
Output
Object File
;ciclo que voleta a r12 y lo guarda en r17
parte_externa_31:
;r12=r31
rol r12
ror r17
dec r20
cpi r20,0x00
brne parte_externa_31
;volvemos a cargar el iterador
ldi r20,0x08
;cilo que voltea a r28 y lo guarda en r18
clc
parte_externa_28:
rol r13
ror r18
dec r20
cpi r20,0x00
brne parte_externa_28
C:\practicasUcontralor\practicas\practica2.asm *
Build
Build Message Find in Files Breakpoints and Tracepoints
I/O View
Name
AD_CONVEF
ANALOG_CO
BOOT_LOAD
CPU
EEPROM
EXTERNAL_
JTAG
PORTA
PORTB
PORTC
PORTD
PORTE
ATmega2560 AVR Simulator Auto Ln 95, Col 18 CAP NUM OVR



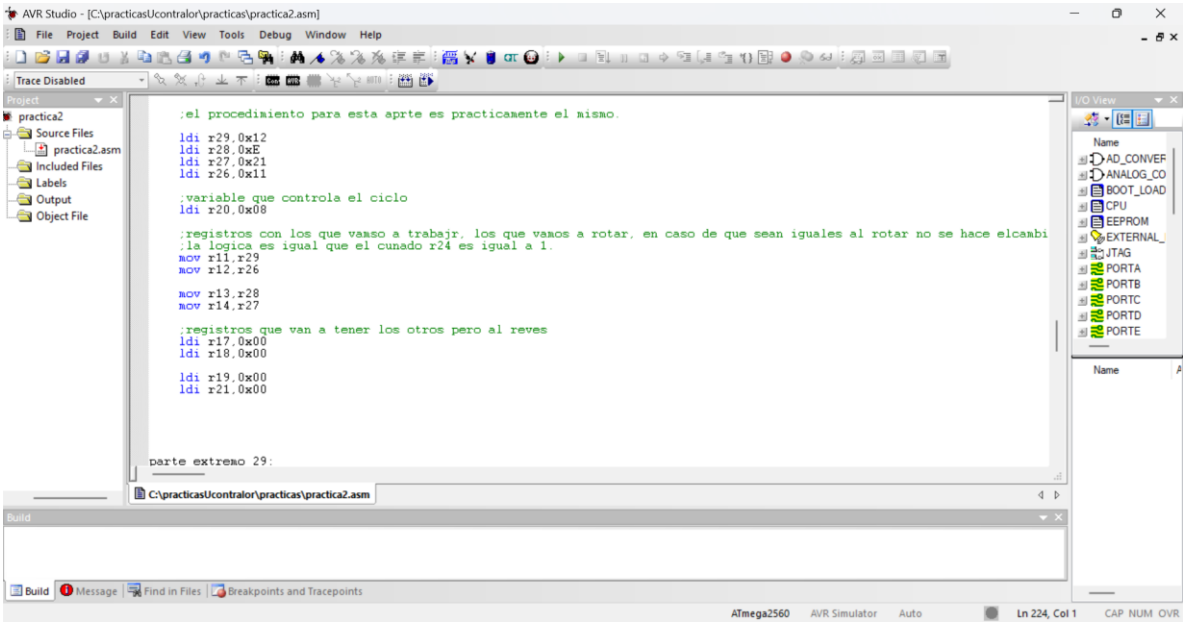
como se menciono con aterioridad, otro requisito es que el programa debería de detectar si un registro al voltearse queda de igual manera que el registro al que se debe intercambiar, esto para no gastar la flash, por lo que se hace uso de una etiqueda simple, que lo único que hace es comparar los 2 registros, el registro ya cambiado y el registro original, con CP, este no modifca los registros solo las banderas. Por lo que lo elegí para la comparación, si la comparación da 0, quiere decir que son iguales por lo que no se hace el cambio, pero si son diferentes si, eso lo hacemos con ayuda de los Branch.

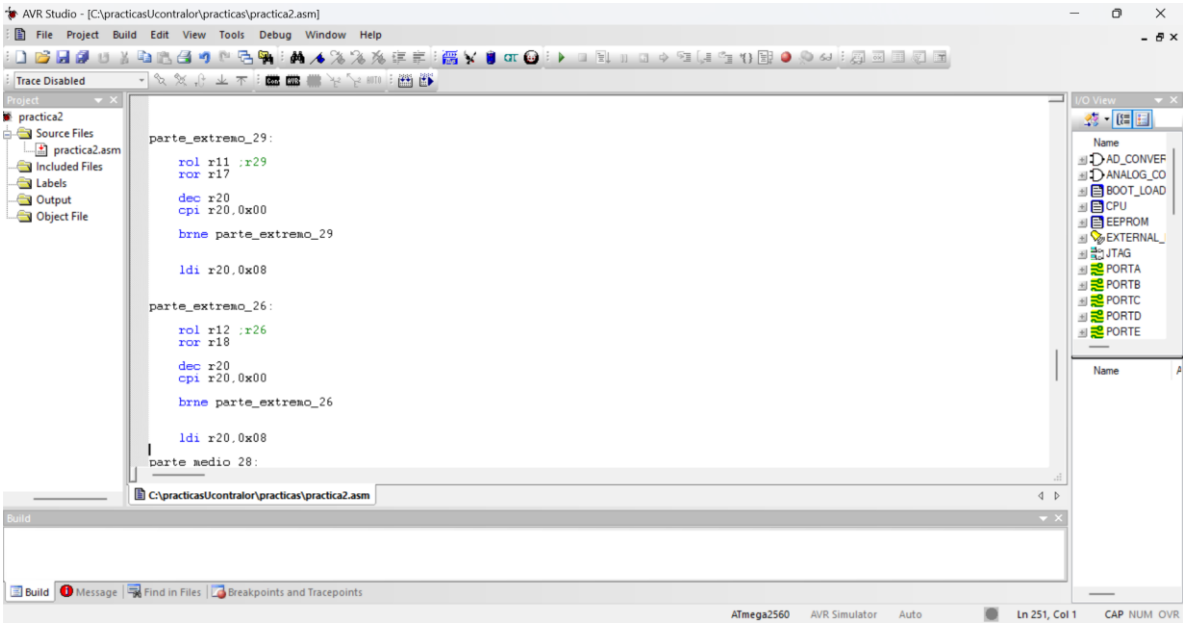


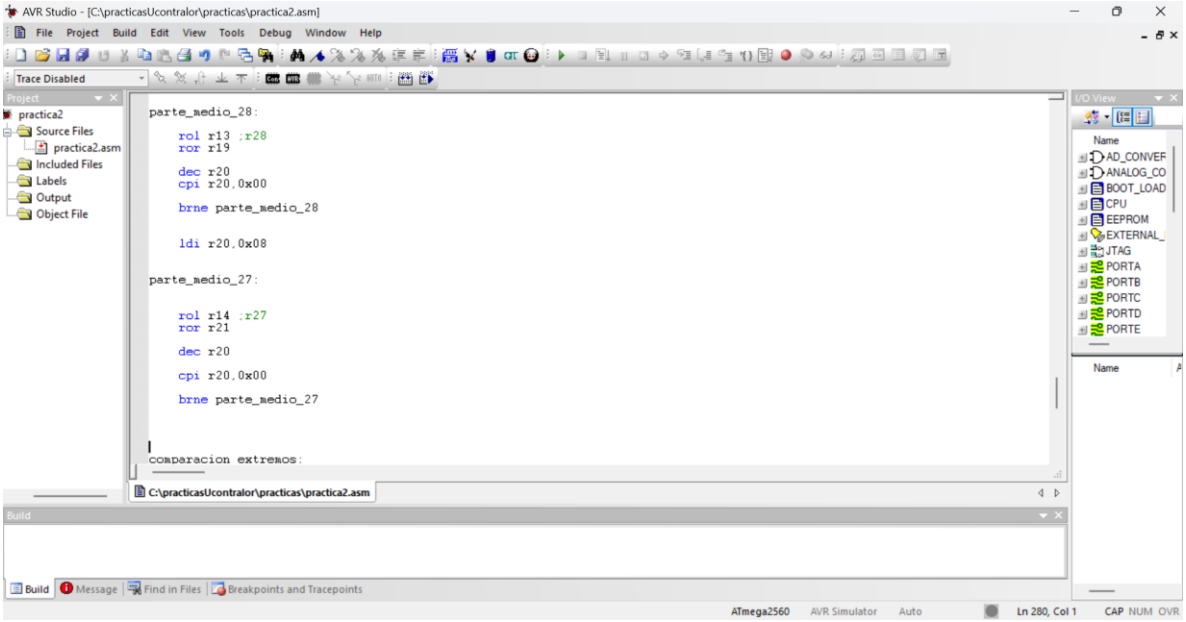


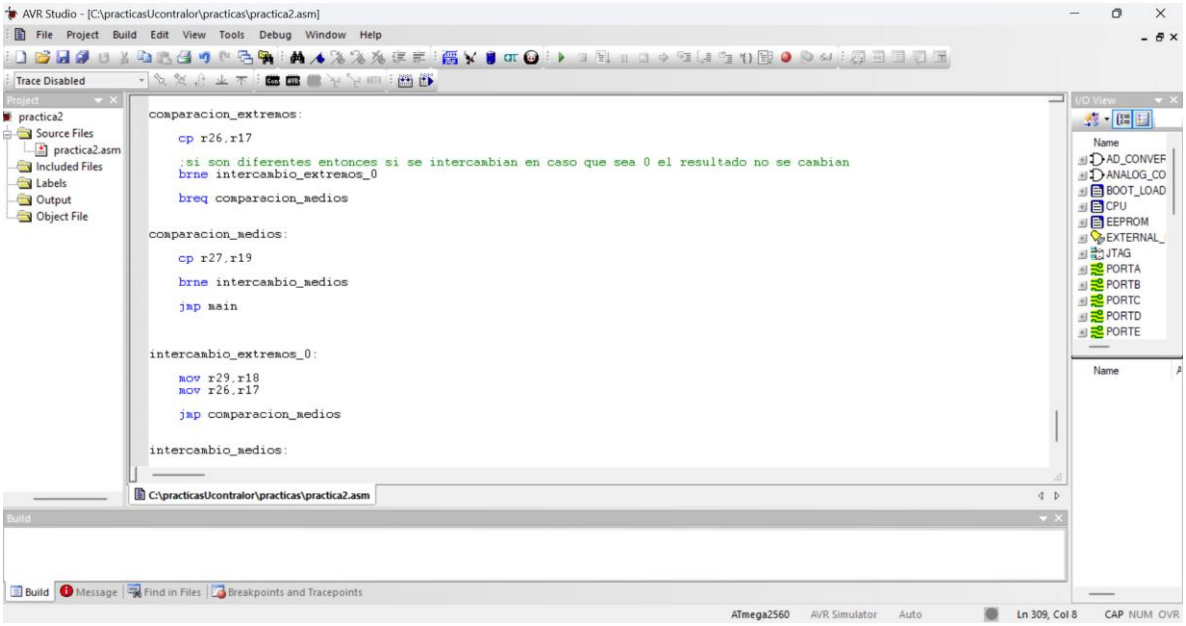
La misma lógica de arriba, se aplica con los registros de cunado R24=0:

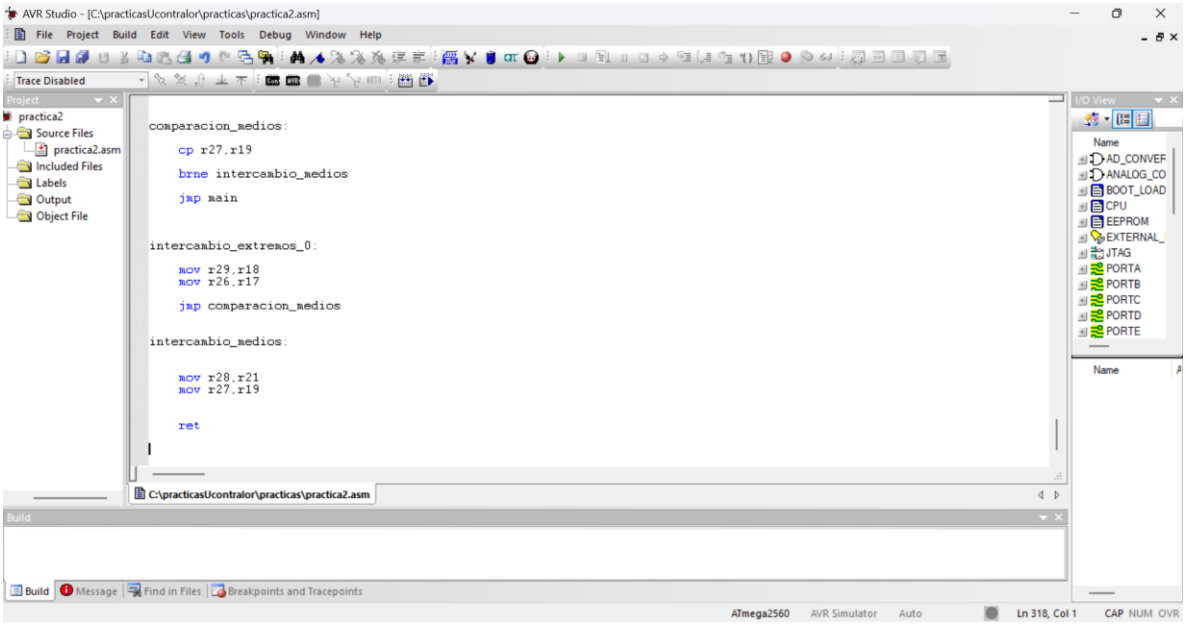
Código:




AVR Studio - [C:\practicasUcontralor\practicas\practica2.asm]
File Project Build Edit View Tools Debug Window Help
Trace Disabled
Project
practica2
Source Files
practica2.asm
Included Files
Labels
Output
Object File
parte_extremo_29:
rol r11 ;r29
ror r17
dec r20
cpi r20,0x00
brne parte_extremo_29
ldi r20,0x08
parte_extremo_26:
rol r12 ;r26
ror r18
dec r20
cpi r20,0x00
brne parte_extremo_26
ldi r20,0x08
parte_medio_28:
C:\practicasUcontralor\practicas\practica2.asm
I/O View
Name
AD_CONVEF
ANALOG_CO
BOOT_LOAD
CPU
EEPROM
EXTERNAL_
JTAG
PORTA
PORTB
PORTC
PORTD
PORTE
Build
Build Message Find in Files Breakpoints and Tracepoints
ATmega2560 AVR Simulator Auto
Ln 251, Col 1
CAP NUM OVR

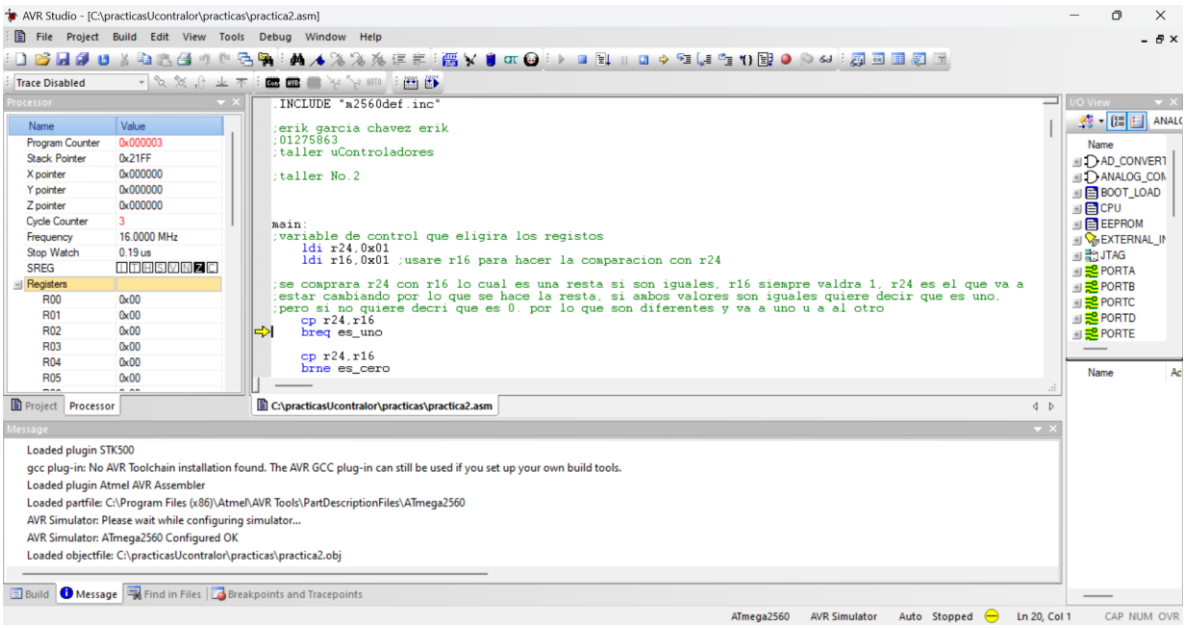

AVR Studio - [C:\practicasUcontralor\practicas\practica2.asm]
File Project Build Edit View Tools Debug Window Help
Trace Disabled
Project
practica2
Source Files
practica2.asm
Included Files
Labels
Output
Object File
parte_medio_28:
rol r13 ;r28
ror r19
dec r20
cpi r20,0x00
brne parte_medio_28
ldi r20,0x08
parte_medio_27:
rol r14 ;r27
ror r21
dec r20
cpi r20,0x00
brne parte_medio_27
comparacion extremos:
C:\practicasUcontralor\practicas\practica2.asm
I/O View
Name
AD_CONVEF
ANALOG_CO
BOOT_LOAD
CPU
EEPROM
EXTERNAL_
JTAG
PORTA
PORTB
PORTC
PORTD
PORTE
Build
Build Message Find in Files Breakpoints and Tracepoints
ATmega2560 AVR Simulator Auto
Ln 280, Col 1
CAP NUM OVR


AVR Studio - [C:\practicasUcontralor\practicas\practica2.asm]
File Project Build Edit View Tools Debug Window Help
Trace Disabled
Project
practica2
Source Files
practica2.asm
Included Files
Labels
Output
Object File
comparacion_extremos:
cp r26,r17
;si son diferentes entonces si se intercambian en caso que sea 0 el resultado no se cambian
brne intercambio_extremos_0
breq comparacion_medios
comparacion_medios:
cp r27,r19
brne intercambio_medios
jmp main
intercambio_extremos_0:
mov r29,r18
mov r26,r17
jmp comparacion_medios
intercambio_medios:
C:\practicasUcontralor\practicas\practica2.asm
I/O View
Name
AD_CONVEF
ANALOG_CO
BOOT_LOAD
CPU
EEPROM
EXTERNAL_
JTAG
PORTA
PORTB
PORTC
PORTD
PORTE
Build
Build Message Find in Files Breakpoints and Tracepoints
ATmega2560 AVR Simulator Auto Ln 309, Col 8 CAP NUM OVR


AVR Studio - [C:\practicasUcontralor\practicas\practica2.asm]
File Project Build Edit View Tools Debug Window Help
Trace Disabled
Project
practica2
Source Files
practica2.asm
Included Files
Labels
Output
Object File
comparacion_medios:
cp r27,r19
brne intercambio_medios
jmp main
intercambio_extremos_0:
mov r29,r18
mov r26,r17
jmp comparacion_medios
intercambio_medios:
mov r28,r21
mov r27,r19
ret
C:\practicasUcontralor\practicas\practica2.asm
I/O View
Name
AD_CONVEF
ANALOG_CO
BOOT_LOAD
CPU
EEPROM
EXTERNAL_
JTAG
PORTA
PORTB
PORTC
PORTD
PORTE
Build
Build Message Find in Files Breakpoints and Tracepoints
ATmega2560 AVR Simulator Auto Ln 318, Col 1 CAP NUM OVR

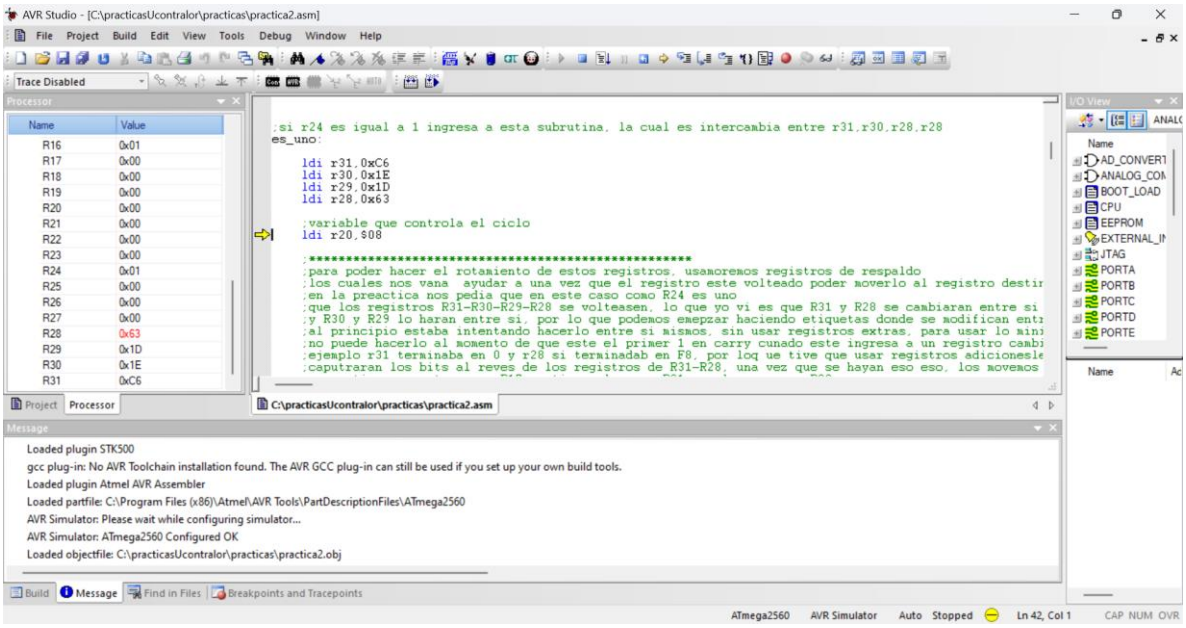
Programa en ejecución:

Tenemos en R24=1.

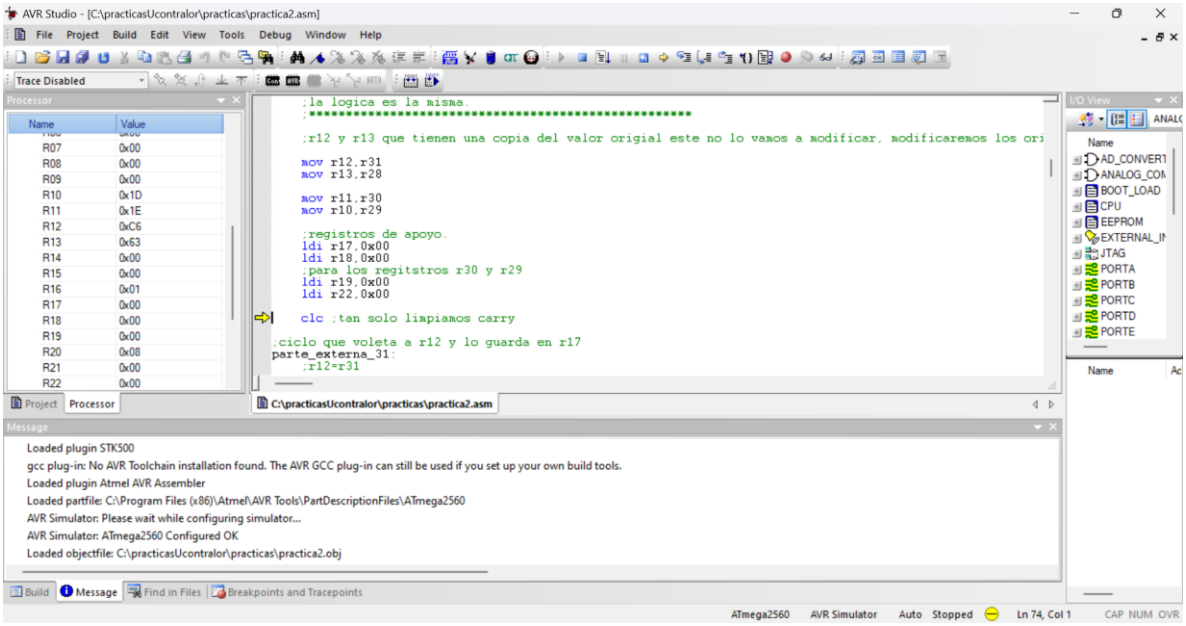


Al hacer la comparación Z se activa, por lo que la operación aritmética-logica fue 0, por lo que se ira ala subrutina es_uno:

Carga los valores en los registros:

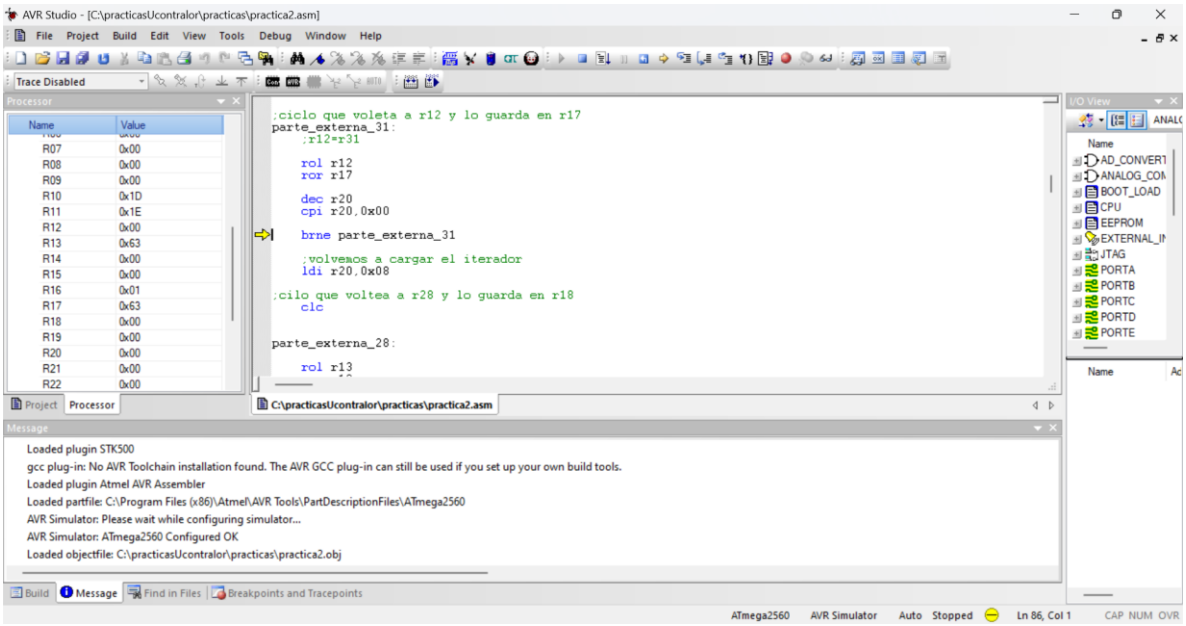


Copiamos los valores en los registros de apoyo e inicializamos los otros igual, limpiamos carry.



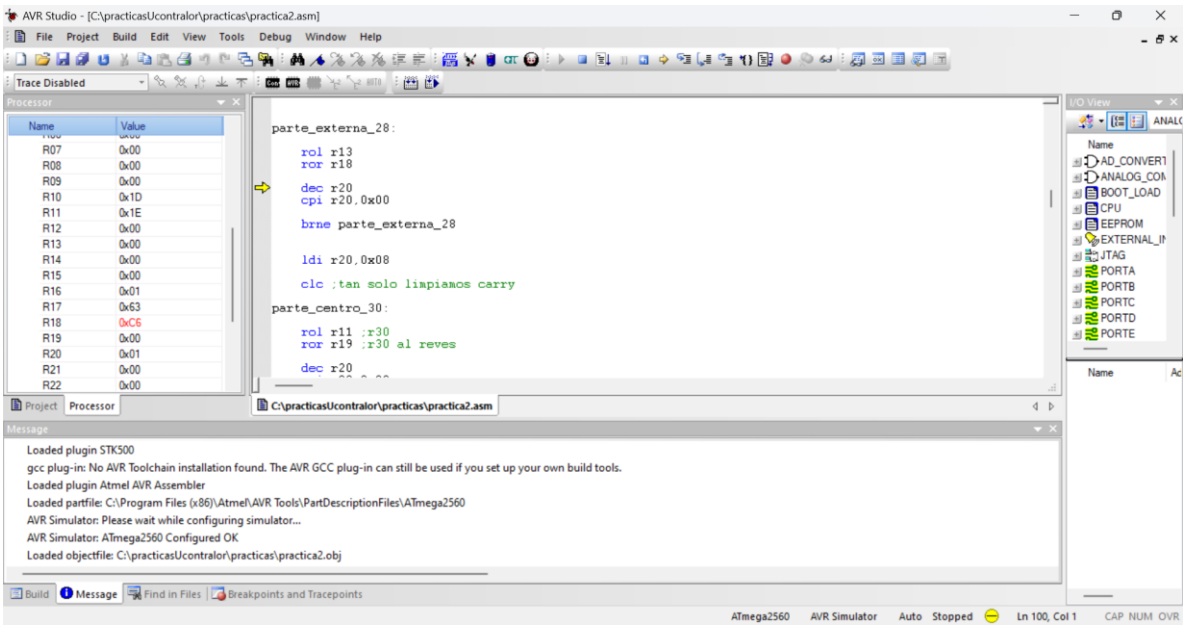
Se empieza a hacer las rotaciones con el primer registro y se guardan en el registro de apoyo que es R17 y el registro que tiene la copia de este es R12

Como se ve R17 tiene ahora 63 que es C6 pero al revés.

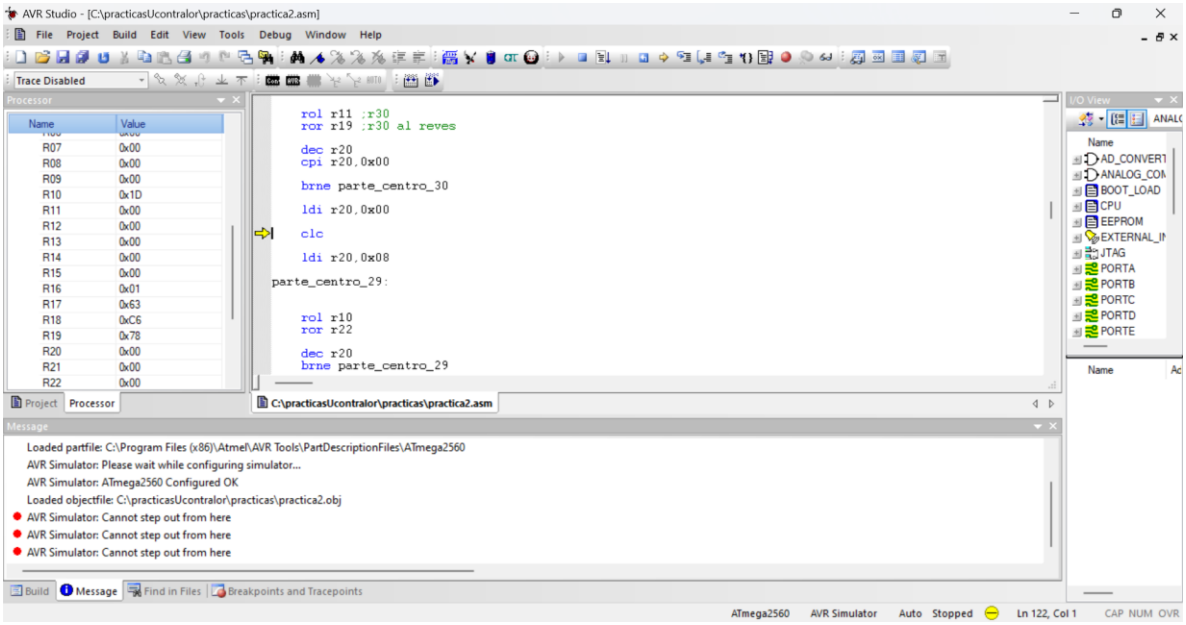


Se hace lo mismo con los otros registros.

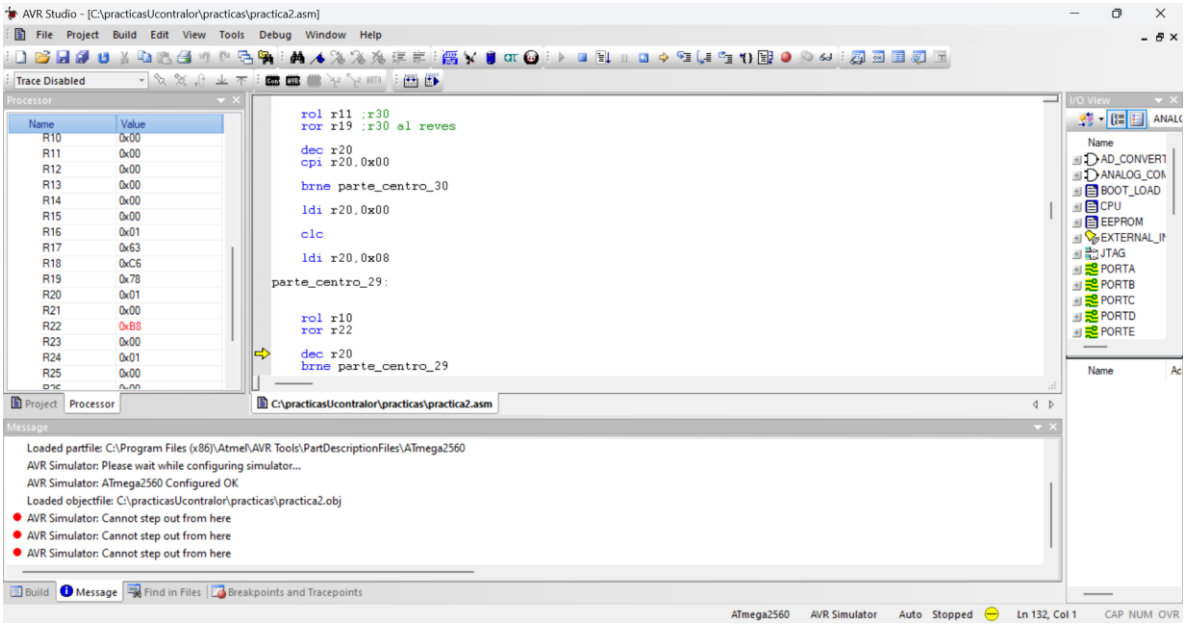
Ahora R18 tiene C6 que es lo mismo de 63, pero al revés.

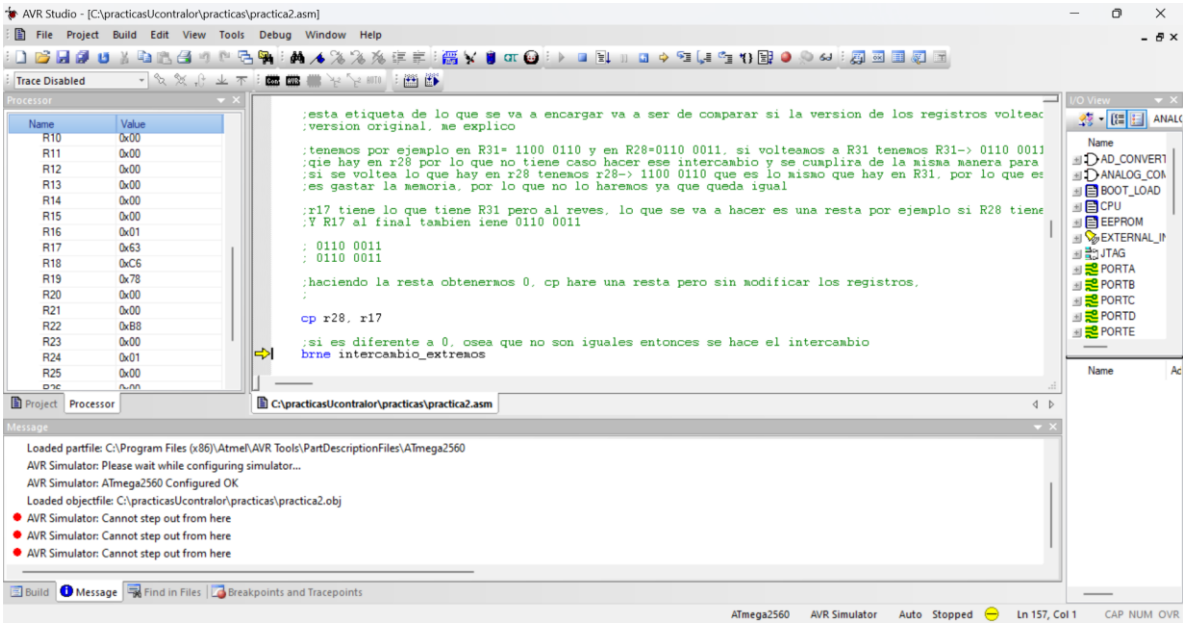


R19 tiene 78 que es lo que tenia R30 = 1E pero al revés.

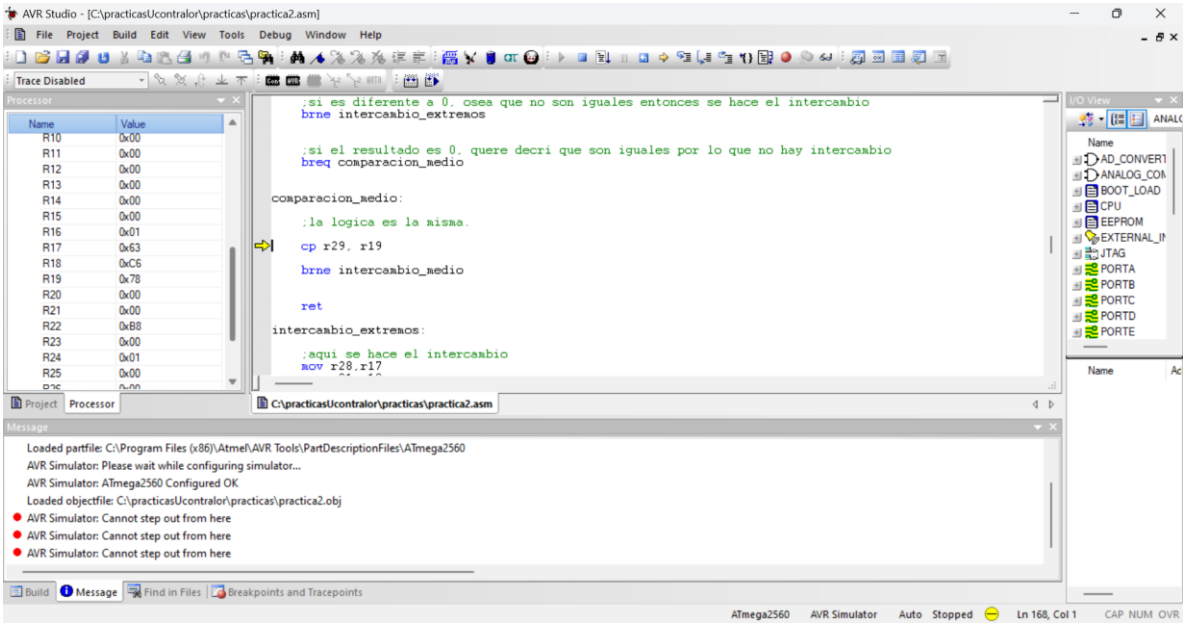


R22 tiene B8 que es lo mismo que tenía R29=1D pero al revés.

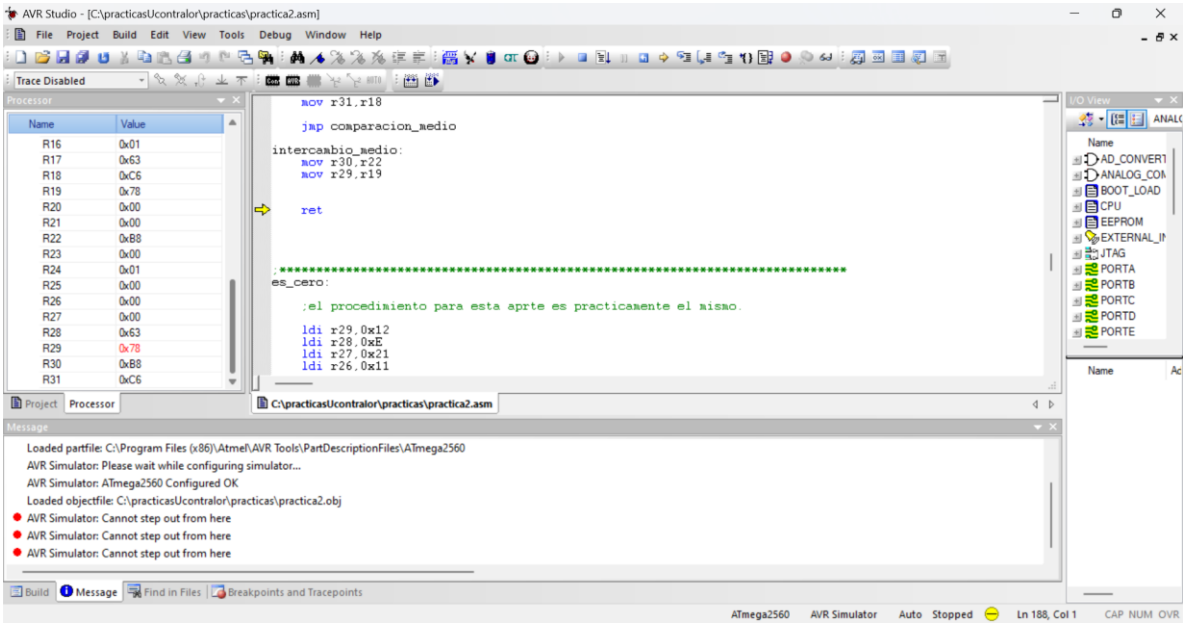




Como R31 y R28 en su froma volteada son lo mismo, entonces no se hace el intercambio, por lo que vamos a ver si los de en medio son iguales o no.



Como no lo son si se hace el cambio.



Dificultades:

La principal dificultad de tuve es sobre como se comportaban las instrucciones, son instrucciones que en efecto sabia el cómo funcionaban en este ensamblador no sabia bien, bien como funcionaba, lo que yo quería era el usar lo mínimo de registros posibles, por lo que estaba buscando el solo hacerlo entre los registros que se debían de intercambiar, pero lo que pasaba era que con ROL, este cunado sacaba el primer 1 con ROR lo que ingresaba era un 0, no el 1, entonces al final el registro R31 quedaba en 00, pero el registro con ROR si quedaba en su forma de al revés, por lo que opte es usar registros de apoyo para poder guardar cada cambio en un registro diferente así evitarme de problemas, pero me tarde bastante en poder encontrar este error, y de igual forma no me quedo del todo claro, a lo que vi, cunado pasa esto la rotación se comporta como un desplazamiento aritmético lógico,

Conclusiones:

Durante la practica pudimos poner en práctica los conocimientos obtenidos en clase, como el manejo de estas nuevas instrucciones para mí, tampoco eran tan desconocidas, lo que hacían si se me eran familiares con respecto al ensamblador del x86, pero hubo unas nuevas para mí, por lo que pude jugar con ellas, y otros tenían un comportamiento que no conocía y eso me ayudo a pensar en como se pudiera solucionar el problema. Pero al final se logró.