Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

Modelación de sistemas mínimos y arquitecturas computacionales

Situación Problema

Soluciona un problema automotriz

Profesora: Nadia Ruth Monfil Campillo

Miranda Magallanes García A00832477

Eduardo Loya Montes de Oca A01383049

Roberto Abraham Pérez Iga A01384237

Alejandro Paredes Balgañón A01351746

Fecha: 24 de octubre del 2021

Los sistemas electrónicos tienen una entrada, procesadores y una salida, los sistemas eléctricos están creados por componentes electrónicos como resistencias, capacitores, transistores, etc... En la industria automotriz la electrónica se usa desde antes de 1970 como en el sistema de frenado.

Los frenos ABS son parte del sistema de estabilidad, comúnmente conocido como control de estabilidad electrónico, el cual monitorea las ruedas durante el frenado brusco. Cada rueda tiene un sensor adjunto, estas se usan para evitar que las llantas del vehículo se traben o inmovilicen, lo que a su vez le da un control mayor al conductor sobre el auto en situaciones de riesgo, evitando así un accidente.

La manera en la que funciona este sistema se basa en la interacción de rotación, traslación y fricción de las llantas al intentar frenar. Estas requieren sumar a 0 para que el auto pueda avanzar exitosamente pero esto cambia cuando frena. La fuerza rotacional para completamente causando un desbalance en las fuerzas el cual resulta en el que el carro siga avanzando en línea recta al frenar.

Aquí es donde entra el sistema ABS y los sensores adjuntos a las llantas. Este sistema libera ligeramente el freno de cada llanta que detecte que se trabe y no permita girar. Esto es para reducir los efectos del frenado, especialmente en terrenos en donde los niveles de fricción varían como las calles nevadas o mojadas.

En este caso, la variable que se utilizará serían la presión detectada de los sensores y esta variable pueden aplicarse en la computadora, siendo la entrada los datos que detecten los sensores, mientras que el proceso sería seguir aplicando una presión óptima de frenado a las ruedas para obtener la salida que sería la estabilidad del vehículo sin bloquear las ruedas, por lo que esto se puede simular en el software de MARIE.js sin ningún problema.

En los automóviles se pueden encontrar hasta 70 u 80 microcontroladores, donde cada uno es considerado en sí, una pequeña computadora con su propio CPU, se adquirió un sistema con un set de instrucciones específico para dar solución a determinados problemas de la industria automotriz y se busca determinar los casos en los que se puede emplear esta computadora hecha a la medida.

Para la situación problema elegimos el caso del frenado ABS, para que funcione este debe cumplirse una condición la cual es gracias a los sensores, por lo que podríamos usar un OR, para expresar que cuando el sensor detecte el bloque en las llantas, se aplique la condición y el proceso.

La siguiente parte de nuestro trabajo se basa en la investigación de información relacionada con los microchips, algunos ejemplares dentro de la industria automotriz, sus características técnicas y cómo estas afectan directamente a su capacidad tecnológica.

En primera instancia investigamos sobre el microchip del modelo PIC18F4550 y sus características técnicas. Se encontró una tabla con especificaciones técnicas de la familia de los microprocesadores PIC18 en donde se encuentra también el PIC18F4550.

Tabla 1. Características de la familia PIC18 (Microchip, 2009).

Features	PIC18F2455	PIC18F2550	PIC18F4455	PIC18F4550
Operating Frequency	DC – 48 MHz	DC – 48 MHz	DC – 48 MHz	DC – 48 MHz
Program Memory (Bytes)	24576	32768	24576	32768
Program Memory (Instructions)	12288	16384	12288	16384
Data Memory (Bytes)	2048	2048	2048	2048
Data EEPROM Memory (Bytes)	256	256	256	256
Interrupt Sources	19	19	20	20
I/O Ports	Ports A, B, C, (E)	Ports A, B, C, (E)	Ports A, B, C, D, E	Ports A, B, C, D, E
Timers	4	4	4	4
Capture/Compare/PWM Modules	2	2	1	1
Enhanced Capture/ Compare/PWM Modules	0	0	1	1
Serial Communications	MSSP, Enhanced USART	MSSP, Enhanced USART	MSSP, Enhanced USART	MSSP, Enhanced USART
Universal Serial Bus (USB) Module	1	1	1	1
Streaming Parallel Port (SPP)	No	No	Yes	Yes
10-Bit Analog-to-Digital Module	10 Input Channels	10 Input Channels	13 Input Channels	13 Input Channels
Comparators	2	2	2	2
Resets (and Delays)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT
Programmable Low-Voltage Detect	Yes	Yes	Yes	Yes
Programmable Brown-out Reset	Yes	Yes	Yes	Yes
Instruction Set	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled
Packages	28-Pin PDIP 28-Pin SOIC	28-Pin PDIP 28-Pin SOIC	40-Pin PDIP 44-Pin QFN 44-Pin TQFP	40-Pin PDIP 44-Pin QFN 44-Pin TQFP

Universal Serial Bus Features:
- USB V2.0 Compliant
- Low Speed (1.5 Mb/s) and Full Speed (12 Mb/s)
- Supports Control, Interrupt, Isochronous and Bulk Transfers
- Supports up to 32 Endpoints (16 bidirectional)
- 1-Kbyte Dual Access RAM for USB
- On-Chip USB Transceiver with On-Chip Voltage Regulator
- Interface for Off-Chip USB Transceiver
- Streaming Parallel Port (SPP) for USB streaming transfers (40/44-pin devices only)

Power-Managed Modes:
- Run: CPU on, peripherals on
- Idle: CPU off, peripherals on
- Sleep: CPU off, peripherals off
- Idle mode currents down to 5.8 µA typical
- Sleep mode currents down to 0.1 µA typical
- Timer1 Oscillator: 1.1 µA typical, 32 kHz, 2V
- Watchdog Timer: 2.1 µA typical
- Two-Speed Oscillator Start-up

Flexible Oscillator Structure:
- Four Crystal modes, including High Precision PLL for USB
- Two External Clock modes, up to 48 MHz
- Internal Oscillator Block:
  - 8 user-selectable frequencies, from 31 kHz to 8 MHz
  - User-tunable to compensate for frequency drift
- Secondary Oscillator using Timer1 @ 32 kHz
- Dual Oscillator options allow microcontroller and USB module to run at different clock speeds
- Fail-Safe Clock Monitor:
  - Allows for safe shutdown if any clock stops

Peripheral Highlights:
- High-Current Sink/Source: 25 mA/25 mA
- Three External Interrupts
- Four Timer modules (Timer0 to Timer3)
- Up to 2 Capture/Compare/PWM (CCP) modules:
  - Capture is 16-bit, max. resolution 5.2 ns (TCY/16)
  - Compare is 16-bit, max. resolution 83.3 ns (TCY)
  - PWM output: PWM resolution is 1 to 10-bit
- Enhanced Capture/Compare/PWM (ECCP) module:
  - Multiple output modes
  - Selectable polarity
  - Programmable dead time
  - Auto-shutdown and auto-restart
- Enhanced USART module:
  - LIN bus support
- Master Synchronous Serial Port (MSSP) module supporting 3-wire SPI (all 4 modes) and I²C™ Master and Slave modes
- 10-bit, up to 13-channel Analog-to-Digital Converter module (A/D) with Programmable Acquisition Time
- Dual Analog Comparators with Input Multiplexing

Special Microcontroller Features:
- C Compiler Optimized Architecture with optional Extended Instruction Set
- 100,000 Erase/Write Cycle Enhanced Flash Program Memory typical
- 1,000,000 Erase/Write Cycle Data EEPROM Memory typical
- Flash/Data EEPROM Retention: > 40 years
- Self-Programmable under Software Control
- Priority Levels for Interrupts
- 8 x 8 Single-Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
  - Programmable period from 41 ms to 131s
- Programmable Code Protection
- Single-Supply 5V In-Circuit Serial

Con estos datos recopilados podemos destacar algunos de mayor importancia como lo serían la dimensión de palabra máxima que puede emular el dispositivo, la cual es de 16 bits, el número máximo de instrucciones u operaciones que se pueden ejecutar dentro del dispositivo el cual es 16,384, la frecuencia máxima de operación del dispositivo que es 48 MHz y el consumo de energía promedio de 0.25 Watts - Voltaje * la corriente.

Estas características después fueron comparadas con otro microchip utilizado dentro de la industria automotriz, nosotros tomamos como comparador al Modelo RH850/P1L-C el cual es mejor en casi todos los aspectos destacados previamente. Tiene 32 bits como su dimensión máxima de palabra, 64KB de RAM y 120 Mhz de frecuencia máxima.

Después de esta comparación se nos encargó crear un programa dentro de MARIE el cual retuviera al menos 50 datos dentro de él. El programa se detalla a continuación:

```
clear /Limpiamos segmentos de memoria
ciclo, Input /Entrada de datos
StoreI Index /Asigna espacio de memoria en el que se va trabajar
Load Index /Se lee valor en la variable Index
Add one /Se suma de la variable uno a la variable index
Store Index /Asignamos resultado a variable index
Load conteo /Cargar contador
subt one /Reducir contador
store conteo /Guardar valor del contador
Skipcond 400 /Si se termino (contador==0) salir del ciclo
Jump ciclo
Halt
conteo, dec 50
Index, HEX 50
one, dec 1
```

Enlace a un video demostrativo de la operación del programa:
https://youtu.be/f2Fs4N7aw5U

El tamaño de este programa es de 520 bits en total.

Gracias a esta investigación nos pudimos dar cuenta de la variedad de distintos microchips que se utilizan en la vida cotidiana y cómo estos se comparan con el programa de MARIE. Dependiendo de su objetivo, son las características que tienen los microchips utilizados a diario. Algunos más poderosos que otros, algunos más baratos y algunos más especializados.

El microchip analizado para la situación problema tiene múltiples usos, incluyendo comunicaciones y la red, electrónica de consumo, multimedia, computadoras, teléfonos, y el procesador RH850/P1L-C se aplica en los sistemas de seguridad de un automóvil. Esto solo nos remarca las diferencias entre los microchips del mercado y sus especializaciones dependiendo del área en el que se usan.

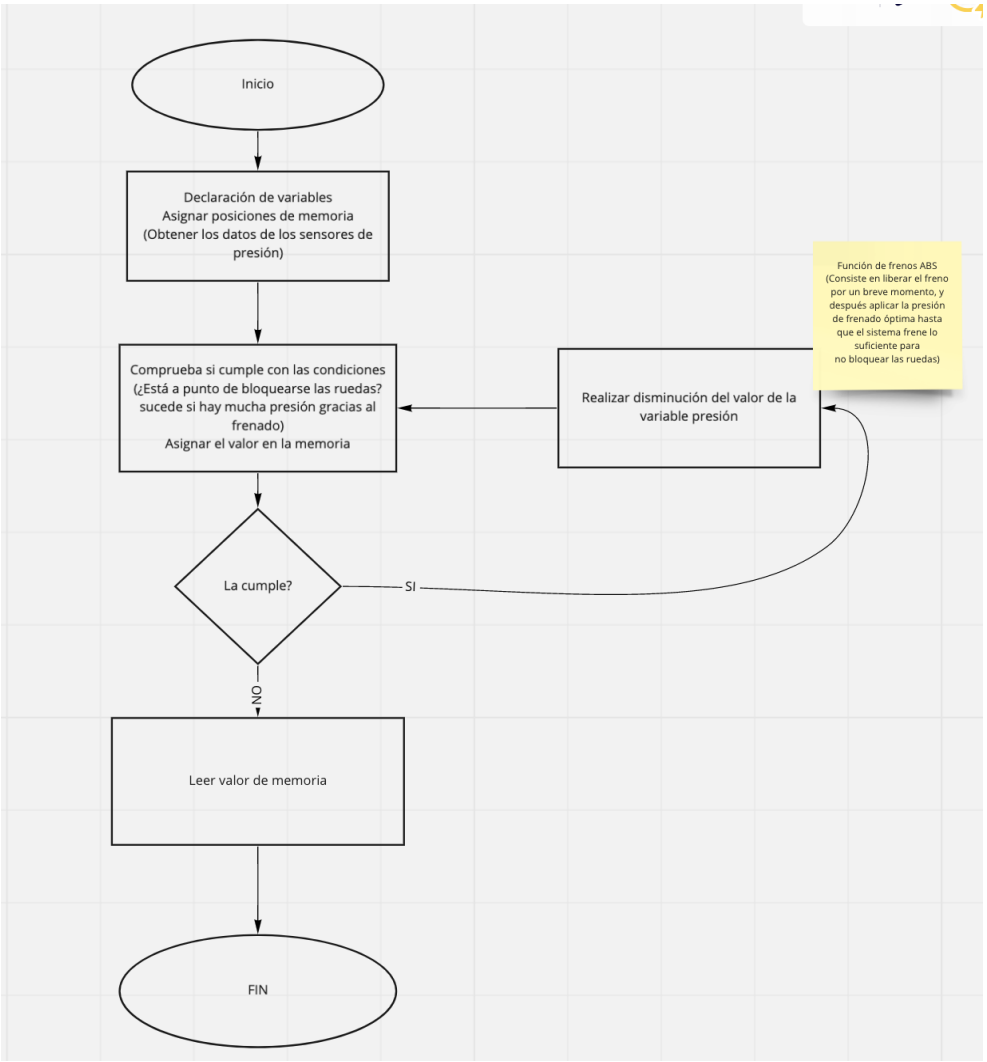
En el programa realizado en MARIE.js pudimos observar el funcionamiento del ensamblador y como esto se relacionaba con la situación problema que elegimos. Contamos con dos sensores en el programa, los cuales van a ser utilizados cuando pongamos de input el 0 o el 1 - así eligiendo qué sensor vamos a utilizar para nuestra prueba, esto se hizo con el objetivo de obtener de él calculo una media estadística de todas las mediciones que se encuentran almacenadas. El programa primero va a checar que la condición skipcond 400 se cumpla con el input del usuario para así pasar al sensor 1, si no es así entonces se va a pasar al sensor 2 para después hacer todo el proceso. En la implementación vamos a tener las variables de los sensores, sus variables para guardar las direcciones, el contador de procesos y al último va a desplegar el resultado de la división.

```
clear

input
skipcond 400        / se salta a la siguiente linea ejecutable si se cumple
jump  sensorOne


sensorOne,  input
            store temp
            load temp
            add ones
            store ones
            storeI dirOne
            load dirOne
            add one
            store dirOne
            load cont
            subt one
            store cont
            Skipcond 400
            jump sensorOne
            jump division1
```

```
sensorTwo,  input
            store temp
            load temp
            add twos
            store twos
            storeI dirTwo
            load dirTwo
            add one
            store dirTwo
            load cont
            subt one
            store cont
            Skipcond 400
            jump sensorTwo
            jump division2

division1,  load ones
            subt cont2
            store ones
            load resultado
            add one
            store resultado
            load ones
            Skipcond 000
            jump division1
            jump end


division2, load twos
            subt cont2
            store twos
            load resultado
            add one
            store resultado
            load twos
            Skipcond 000
            jump division2

end,        Output
            halt
```

```
ones, dec 0     / variable para primer sensor
twos, dec 0     / variable para segundo sensor
dirOne, hex 040 / variable para guardar las direccion 1
dirTwo, hex 100 / variables para guardar las direccion 2
/ el contador de procesos
cont, dec 5
cont2, dec 5
temp, dec 0
one, dec 1
/ resultado
resultado, dec 0        / resultado de la division
resultado2, dec 0       / resultado de la division 2
```

Por último, hicimos un diagrama de flujo para representar la posible solución al sistema de frenado ABS:



La diferencia que tendría este programa con los anteriores es que se tomaría en cuenta un sensor que represente el valor de la presión de frenado, esto con la finalidad de conocer si las llantas están a punto de bloquearse, si es así, se aplicará un método para disminuir el valor de la presión, y volver a comprobar si las llantas están a punto de bloquearse, si no es el caso, simplemente va a leer el último valor guardado dando a entender que la presión no es tan alta para requerir el sistema de frenado ABS.

El primer programa se puede usar para almacenar los datos de la presión en cada ciclo que se haga, es decir que al guardar estos datos se pueda analizar cómo fueron cambiando conforme el sistema ABS era aplicado.
Mientras que el segundo programa es muy parecido a este problema, ya que detecta los valores por medio de sensores, y después realiza un división (que viene siendo restas), por lo que se puede usar en la parte de la disminución de la presión, ya que el objetivo es disminuir la presión detectada por el sensor.

En conclusión, el sistema electrónico del vehículo es lo más parecido a un cerebro que tiene un automóvil. Recibe información, la procesa y envía las instrucciones precisas a cada uno de los sistemas del vehículo. Sabemos que el lenguaje ensamblador es muy útil para resolver distintos problemas en la industria automotriz, ya que utilizan muy poca memoria y su uso más común es la programación de microcontroladores, el cual es un circuito integrado programable que está compuesto de varios bloques funcionales que cumplen una tarea específica. Es por eso que el lenguaje ensamblador puede ayudar a resolver los principales problemas que tienen algunos vehículos en su electrónica, como lo puede ser en el frenado ABS, en la inyección de combustible o en el sistema de enfriamiento.