KARADENİZ TEKNİK ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
MİKROİŞLEMCİLER LABORATUVARI

İkili Tabanda Çarpma

1. GİRİŞ

Mikroişlemciler, bilgisayar sistemlerinin temel bileşenleri olarak toplama ve çıkarma gibi aritmetik işlemleri gerçekleştirmek üzere özel talimat (instruction) setlerine (örneğin, ADD ve SUB) sahiplerdir. Bunun yanında, modern mikroişlemciler bu temel aritmetik işlemlerin üzerine ek olarak çarpma ve bölme fonksiyonunu sağlayan özel bir donanım birimine (örneğin, ALU içinde bir çarpma ünitesi) sahiptir.

Ancak; Motorola 6800, MOS Technology 6502 ya da Intel 8085 gibi eski ya da halihazırda günümüzde üretilse de düşük maliyetli projelere hizmet eden mikroişlemcilere bakıldığında bu özel donanım birimlerine yer verilmemiş olup bahsi geçen çarpma ve bölme işlemleri donanım tabanlı gerçekleştirilememektedir. Donanımın el vermediği durumda yazılım ile kompanse edilen birçok durum gibi, bu durumda da mikroişlemcinin bir donanım yoksunluğu (özel bir çarpma birimi) yazılımsal olarak toplama ve kaydırma (shift) gibi daha basit komutlar ile bir döngü içerisinde gerçekleştirilebilmektedir.

2. DENEY AMACI VE KAZANIMI

Deneyde, kullanılacak mikroişlemcinin Assembly dili üzerinde Shift-and-Add algoritması kullanılarak işaretsiz iki sayısal veri üzerinde yazılımsal bir çarpma işlemi gerçekleştirilecektir.

Bu deneyde, x86 mimarisinin ilk üyesi olan Intel 8086 mikroişlemcisi kullanılacaktır. Bahsi geçen mikroişlemcilerden farklı olarak Intel 8086'da çarpmaya özel MUL/IMUL emri bulunuyor olsa da Intel 8086, günümüzde yaygın olarak kullanılan Intel ve AMD işlemcilerin temelini oluşturan x86 komut setinin başlangıç noktasıdır. Bu durum, katılımcıların kendi bilgisayarlarında benzer komut setiyle uygulama yapabilmelerine olanak tanımaktadır. Ayrıca, 8086'nın nispeten basit ve anlaşılır bir komut yapısına sahip olması, mimarinin öğrenilmesini kolaylaştırmaktadır.

Gerçekleştirilen bu deney sayesinde proje maliyetinin düşük olduğu, güç tüketiminin düşük olmasının istendiği ya da mikroişlemci boyutunun düşük olması istendiği durumlarda çarpma için özel bir donanıma yer verilmeyen günümüz mikroişlemcilerinde (örneğin, ATtiny4/5/9/10 ya da Microchip'in PIC10F serisi) çarpma işlemi gerçekleştirilebilecektir. Bu yaklaşım, donanım-yazılım dengelemesini sağlamaya yönelik bir mühendislik ile gerçekleştirilecek olup ilgili kazanımın deney katılımcılarına kazandırılması hedeflenecektir.

3. DENEYE HAZIRLIK

Deneyden istenilen verimin alınabilmesi ve deney kazanımının kazanılabilmesi için deneye gelmeden önce incelenmesi gereken konular şu şekildedir:

- Yazılan bir kodun mikroişlemci için çalıştırılabilir bir program haline gelirken geçtiği adımlar ve neden Assembly'ye çevrildiği incelenmelidir.
- Bahsi geçen Motorola 6800, Intel 8085 ve Intel 8086 mikroişlemcilerinin Assembly emir setleri karşılaştırılmalı, farklar incelenmelidir.
- Intel 8086 mikroişlemcisinin emir setinde bulunan ADD, MOV, RET gibi temel mnemonic'ler hakkında bilgi sahibi olunmalı, "godbolt.org" sitesi kullanılarak deneysel ve interaktif bir şekilde örnek bir C kodu yazılıp kodun Assembly karşılığı gözlenmelidir.
- Mikroişlemcilerde bulunan register'ların Assembly dilinde ne amaçla ve nasıl kullanıldığı incelenmelidir. Bir veri değişkeni (variable) tanımlandığında bu değişkenin nasıl ve nerede tutulduğunu inceleyiniz. Bu amaç doğrultusunda "godbolt.org" web sitesinden faydalanılabilmektedir.
- C programlama dilinde bulunan "signed" ve "unsigned" kavramları ve farkları incelenmelidir. Bu kavramların sayısal veriler üzerine tanımlandığı "int" ve "uint" veri türleri ile bu veri türlerinin bulunduğu "short (u)int", "(u)int" ve "long (u)int" veri türlerinin bellek üzerindeki farkları incelenmelidir.
- "Deneyin Yapılışı" başlığında verilmiş olan ve Intel 8086 mikroişlemcisi üzerinde gerçeklenecek Shift-and-Add yazılımsal çarpma işlemine dair algoritma incelenmeli ve anlaşılmalıdır.

4. DENEYİN YAPILIŞI

Assembly üzerine bilgileri tazeleme amaçlı C programlama dilinde temel aritmetik işlemler ve veri tanımlamalarına sahip basit düzeyde bir program derlenecektir. Bu programın, bir disassembler programı aracılığıyla (örneğin, IDA, Binary Ninja, Ghidra) mikroişlemcide karşılığı olan Assembly kodu ve program akışı gözlenecektir. Gözlenen bu emirler düşük seviye bir hata ayıklama (debugger) aracıyla (örneğin, x86dbg ya da gdb gibi) tek tek koşularak işlemci yazmacındaki (register) değişiklikler ve verilerin hafızada nasıl tutuldukları incelenecektir.

Deneyde C dilinde işaretli ve işaretsiz "short int", "int" ve "long int" sayısal veri türleri oluşturularak bu veri türlerinin çeşitli mikroişlemcilerdeki Assembly kodu karşılıkları incelenerek derleyicinin ve hedef mikroişlemcinin farklılıkları gözlenecektir. Oluşturulan veri türleri ile çarpma işlemi gerçekleştirilip sonradan kıyaslamak üzere çarpma emri bulunan modern mikroişlemcilerdeki makine kodu incelenecektir.

Intel 8086 mikroişlemcisinin Assembly dilinde yazılarak koşulacak olan ve işaretsiz iki sayısal verinin çarpımı üzerine kullanılacak olan Shift-and-Add algoritması ile akış diyagramı (flowchart) aşağıdaki gibidir:

Shift-and-Add Algoritması

1. **(Başlama)**
   Çarpan değeri 0'a eşitlenir. Çarpan ve Çarpılan sayıları belirlenir.
2. **(Kontrol ve Ekleme)**
   Çarpan sayısının en sağdaki (en anlamsız) bitine (LSB - Least Significant Bit) bakılır:
   Eğer LSB 1 ise Çarpılan sayı değeri o anki Sonuç değerine eklenir.
   Eğer LSB 0 ise, toplama işlemi yapılmaz.
3. **(Kaydırma)**
   Çarpılan sayı bir bit sola kaydırılır (<< 1). Bu onluk tabandaki çarpmada bir sonraki basamağa geçmeye benzer.
   Çarpan sayı bir bir sağa kaydırılır (>> 1). Bu bir sonraki biti kontrol etmek için yapılır.
4. **(Döngü ile Çarpmayı Tamamlama)**
   Bu işlemler Çarpan sayısının tüm bitleri kontrol edilene kadar (örneğin, 32-bitlik sayılar için 32 kez, 8 bitlik sayılar için 8 kez) tekrarlanır.
   Döngü tamamlandığında Sonuç değişkeninde çarpma işleminin nihai sonucu elde edilmiş olur.

Shift-and-Add Algoritmasının Akış Diyagramı