HCS12 EmumiS: Sanal HCS12 Geliştirme Aracı
HCS12 EmumiS: Virtual HCS12 Development Tool

Hüseyin Ulutaş, Alp Arslan Bayrakçi

Bilgisayar Mühendisliği
Gebze Yüksek Teknoloji Enstitüsü, Kocaeli
ulutashus@gmail.com, bayrakci@bilmuh.gyte.edu.tr

Özet

Mikrodenetleyici programlamada gelişmek için geliştirme kartları kullanılmaktadır. Ancak maalesef, özellikle eğitim alanında çok popüler olan HCS12 mikrodenetleyicilerinin geliştirme kartları hem yurt içinden temin edilememekte hem de yüksek fiyatları sebebiyle çoğu kimse tarafından alınamamaktadır. Biz bu bildiride tamamen bir bilgisayar yazılımı olan ancak hem HCS12 benzetimi, bellek değerleri gösterimi, hata ayıklama özelliklerine sahip hem de üzerinde LED'ler, 7-segment LED, LCD, vızlayıcı (buzzer) ve dip-anahtarlar bulunan özelleştirilebilir bir geliştirme kartına öykünebilen HCS12 EmumiS isimli sanal bir HCS12 Geliştirme Aracı tasarladık.

Abstract

For learning and development purpose, microcontroller evaluation boards are widely used. However, the difficulty to get HCS12 boards domestically and also the expensiveness of them make it impossible to get for most of the students and hobbyists although HCS12 is very popular in education. In this paper, we present HCS12 EmumiS, which is a computer software that can not only simulate HCS12 instructions, memory and debug the program but also can fully emulate a customizable evaluation board, having components like LEDs, LCDs, 7-segment LEDs, buzzers and dip switches.

1. Giriş

Freescale firmasının Eylül 2004'te tanıttığı HCS12 mikrodenetleyici ailesi özellikle eğitim alanında çok popüler olan Motorola'nın 68HC11 serisinin 16-bit adreslemeyi destekleyen sonraki neslidir. HCS12, özellikle eğitim, otomotiv ve kontrol sektöründe yaygın kullanılmaktadır [1]. CISC (Complex Instruction Set Computer) mimarisine sahip olan HCS12 mikrodenetleyici ailesi SCI (UART), SPI, I2C, CAN iletişim protokollerini destekleyen modüllerin yanı sıra gelişmiş kesme sistemi, zamanlayıcı ve PWM modüllerini içerir [2].

University of California [3], Carnegie Mellon University [4], University of Texas [5], Georgia Institute of Technology [6] gibi onlarca üniversite ve enstitüde, gömülü sistemler ve mikrodenetleyiciler gibi derslerde bu mikrodenetleyici ailesi öğretilmektedir. Ayrıca HCS12 üzerine, lisans ve yüksek lisans düzeyinde eğitim amaçlı birçok kitap bulunmaktadır [1, 7-9].

Mikroişlemci ve mikrodenetleyici programlama alanında gelişmek için üzerinde LCD, LED'ler, anahtarlar, vızlayıcı (buzzer) vb. çevre birimleri bulunan geliştirme kartları kullanılmaktadır. Öğrencilerin mikroişlemciler ve gömülü sistemler gibi derslerde kendilerini geliştirebilmeleri için de bu geliştirme kartlarına sahip olmaları gerekmektedir. Ancak, yüksek fiyatları ve yurt içinden temin edememe zorluğu gibi nedenlerle Dragon12 Plus [10] ve Freescale LFEBS12UB [11] gibi HCS12 geliştirme araçlarına bireysel olarak sahip olmak oldukça zordur. Sonuç olarak, sadece laboratuar ortamında kullanılabilen geliştirme kartları yüzünden eğitim alanında çok önemli bir yer tutan HCS12 mikrodenetleyicisinin eğitiminden gereken verim alınamamaktadır.

Bu durum, ücretsiz olarak elde edilebilen Freescale firmasının CodeWarrior [12] benzetim yazılımı ve SIMHCS12 [13] gibi HCS12 benzeticilerinin (simülatör) kullanılmasıyla aşılamaz. Çünkü bu benzeticiler sadece HCS12'nin iç yazmaç (register) ve bellek içeriklerindeki değişimi göstermektedir. Kesmeler, gecikmeler, LCD ve LED kullanımı, vızlayıcı kullanımı, SCI gibi iletişim modüllerinin kullanımı, mikrodenetleyici bacaklarındaki sinyallerin görüntülenmesi, kullanıcıya bağlı anahtar girdisi kullanımı gibi geliştirme kartıyla test edilebilecek senaryoların benzetimini (simulation) ve öykünmesini (emulation) yapamamaktadır.

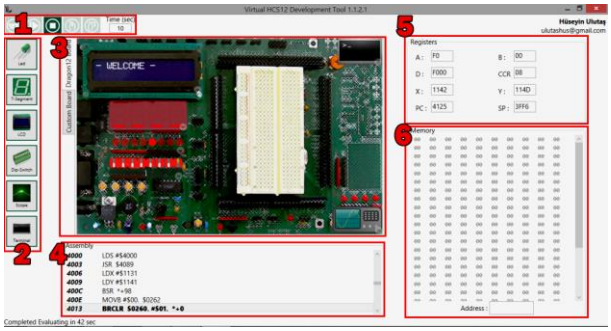
Bu çalışmada; HCS12 için yazılan C, C++ veya Assembly koduna göre hem HCS12 iç yazmaçlarının ve bellek değerlerinin benzetimini hem de Dragon12 geliştirme kartı [10] gibi üzerinde LCD, LED'ler, vızlayıcı, 7 segment LED'ler, anahtarlar bulunan bir geliştirme kartının öykünmesini yapabilen sanal HCS12 geliştirme aracı tasarladık. HCS12 EmumiS adını verdiğimiz bu yazılımsal araç, aynı zamanda PC'nin seri portu üzerinden SCI iletişimi benzetimi de yapabilmektedir. HCS12 EmumiS sayesinde kullanıcılar ne HCS12 mikrodenetleyicisine ne de herhangi bir geliştirme kartına ihtiyaç duymadan, HCS12 programları yazıp çalıştırma, ayıklama ve gerçek zamanlı benzetim ve tamamen kullanıcı tarafından özelleştirilebilen geliştirme kartı

öykünme sonuçlarını gözlemleme imkânı bulabilmektedir. Bu kadar gelişmiş ve özelleştirilebilir bir benzetim ve öykünme aracı bildiğimiz kadarıyla HCS12 literatüründe mevcut değildir.

2. HCS12 EmumiS

HCS12 mikrodenetleyici benzeticisi ve geliştirme aracı öykünücüsü görevi gören HCS12 EmumiS çalışırken alınmış bir arayüz görüntüsü Şekil 1'de mevcuttur. Uygulama WPF ve .NET teknolojileri kullanılarak C# programlama diliyle geliştirilmiştir. Geliştirme esnasında harici olarak ücretsiz olan OxyPlot [15] kütüphanesi ve CodeWarrior'ın çözümleyicisi[1] [12] kullanılmıştır.

Şekil 1'de HCS12 EmumiS arayüzü ve kırmızı dikdörtgenlerle ayılmış bölmeler yer almaktadır. Şekilde yer alan Bölme 2'deki tamamen özelleştirilebilir altı geliştirme kartı bileşenini istediği özelliklerde ve istediği HCS12 bacağına bağlayarak, kullanıcı kendi geliştirme kartını inşa edebilmektedir. Bölme 6'da tüm HCS12 bellek değerlerini gözlemleme ve değiştirebilme, Bölme 5'te ise HCS12 iç yazmaçlarını izleme ve değiştirebilme imkânları sunulmaktadır. Bölme 4, kullanıcının EmumiS tarafından çalıştırılan Assembly komutlarını takip etmesini sağlar. Bölme 3'te ise kullanıcı kendi özelleştirebildiği geliştirme kartını üzerindeki dinamik bileşenlerle birlikte gözlemlemektedir.



Şekil 1: HCS12 EmumiS Arayüz Görüntüsü.

2.1. HCS12 EmumiS Yetenekleri

Burada uygulamanın sahip olduğu yetenekleri listeleyeceğiz:

1. HCS12 Komut Kümesi Benzetimi:
   - Komut kümesinin %94'ünün (196/208 farklı komut) desteklenmesi.
   - Zamanlama sisteminin benzetimi (TCNT sayacı, ön-çarpıcı, Girdi yakalama, Çıktı Kıyas ve Gerçek Zaman özellikleri).
   - SCI sisteminin benzetimi (Bu sayede UART kullanan herhangi bir fiziksel ürün bilgisayarın seri portu üzerinden HCS12 EmumiS ile haberleştirilebilir).
2. Desteklenen kesmeler:
   - Zamanlayıcı taşma (Timer Overflow) kesmesi.
   - Girdi Yakalama (Input Capture) kesmesi.
   - Çıktı Kıyas (Output Compare) kesmesi.
   - SCI gönderme ve alma (TxRx) kesmeleri.
   - Gerçek Zaman kesmesi (Real Time Interrupt (RTI)).
3. Desteklenen geliştirme kartı bileşenleri:
   - LED bileşeni.
   - 7-Segment LED bileşeni.
   - LCD bileşeni. (16x2 bloklu)
   - Dip-Anahtar bileşeni.
   - Vızlayıcı (buzzer) bileşeni
   - Osiloskop bileşeni. (Herhangi bir bit veya baytı izleme)
   - Terminal bileşeni. (SCI sistemine bağlı)
4. Esneklik:
   - LED, 7-Segment, LCD, dip-anahtar, osiloskop ve terminal bileşenlerinin istenen sayıda, istenen özelliklerde ve istenen HCS12 bacağına bağlanabilmesi.
   - Yeni bileşen ekleme, kaldırma, taşıma, özelliklerini değiştirebilme imkânları.
   - Assembly, C ve C++ programlama dillerinde yazılmış programların desteklenmesi.
   - Yüklü olan programda yapılan değişiklerin otomatik algılanması.
5. Hata ayıklama yetenekleri (debug):
   - Assembly seviyesinde adım adım çalıştırabilme.
   - Duraksatma noktaları (breakpoint) ile istenilen yere kadar çalıştırma
   - Bellek değerlerini gözlemleme ve değiştirme imkânı.

2.2. Koddan Komut Ayrıştırmaya

Assembly, C veya C++ programlama dilleriyle girdi olarak verilen program önce derlenerek s19 uzantılı HCS12 çalıştırılabilir dosyasına dönüştürülüyor. Daha sonra EmumiS bu s19 dosyasıyla CodeWarrior çözümleyicisini çağırarak Şekil 2'deki gibi standart bir Assembly kodu üretiyor. Bu koddan metin ayrıştırmasıyla komutları ayırarak her bir komutu aynen HCS12 gibi işletiyor. Ayrıca komutların kaç çevrim (cycle) süreceği bilgisi de şekilde görüldüğü gibi bu dosyada mevcut.

```
Adres     Hex      Cycle  Opcode  Operand
00004000 CF4001    [2]    LDS    #16385
00004003 0711      [4]    BSR    *+19
00004005 CE3000    [2]    LDX    #12288
00004008 A630      [3]    LDAA   1,X+
0000400A 164076    [4]    JSR    $4076
0000400D 1640A1    [4]    JSR    $40A1
00004010 8E300C    [2]    CPX    #12300
00004013 26F3      [3/1]  BNE    *-11
00004015 3D        [5]    RTS
```

Şekil 2: Tüm girdilerin dönüştürüldüğü Assembly kodunu içeren dosyadan bir kesit

HCS12 komut kümesinde 208 farklı makine koduna sahip özgün komut bulunmaktadır. Bu komutlar yedi farklı adresleme seçeneğinden birine girebilmekte, farklı sayıda işlenen (operand) alabilmektedir. Hiç işlenen almayan komutlar bulunabildiği gibi farklı adresleme seçeneklerinde dörde kadar işlenen alan komutlar da bulunmaktadır. Başka bir problem de komutların kaç çevrim (cycle) süreceğinin

[1] HCS12 makine dilinden HCS12 Assembly'ye dönüştürücü

hesaba katılarak gerçek zaman mefhumunun takip edilebilmesidir.

Şekil 2'deki gibi bir girdi dosyası anlamlandırılırken ilk olarak komutların sayısal makine kodu değerleri ilgili bellek adreslerine yüklenir böylelikle bellek benzetime hazır hale gelir. Bunun dışında her bir komutun nesneleri oluşturulur ve bu nesne, komutun kendi metoduyla ilişkilendirilir. Her bir komutun kaç çevrim süreceği Şekil 2'deki gibi bir girdi dosyasından elde edilerek gerçek zaman takibinde bu bilgi kullanılır. Komutların işlenenlerinin sayısı ve ne olduğu da EmumiS tarafından bu ayrıştırma esnasında belirlenir.

2.3. Komut Benzetim Performansı İyileştirmeleri

Mikrodenetleyicinin çalışma benzetiminde performans çok kilit bir rol oynamaktadır. Günümüz genel amaçlı bilgisayarlarının işlem güçleri HCS12 mikrodenetleyicisine göre yaklaşık yüz kat daha fazla olmasına rağmen bilgisayarda benzetim performansı gerçek hayattaki HCS12 çalışma performansından daha yavaştır. Bunun sebepleri; ayrıştırmada kaybedilen zaman, üst seviye bir dille bunların yapılması, geliştirme kartı bileşenlerinin görsel özelliklerini bilgisayar ekranında göstermek için harcanan işlem gücü ve bilgisayarın kaynaklarını bize ayırmaması olarak sıralanabilir.

Bu sorunların hepsini çözmek mümkün olmasa da ayrıştırma daha akıllı hale getirilebilir. Örneğin, Şekil 2'deki "LDAA" komutunun parametresi metinde görüldüğü gibi "1,X+" şeklinde kabul edilirse komut döngü içerisinde her çalıştırıldığında, benzetici bu karakter dizisini tekrar tekrar anlamlandırmaya ihtiyaç duyacaktır. Bu da binlerce tekrarlı döngülerde çok ciddi performans kaybına yol açmaktadır. Bunun yerine, HCS12 EmumiS ayrıştırma yaparken her komut için sadece bir defa anlamlandırma yaparak komutun, adresleme seçeneğinin ve işlenenlerinin ne olduğunu öğrenip bunu aklında tutan bir yapıda tasarlandı. Elbette bunu yaparken döngünün her bir tekrarında komutların farklı işlenenlerle çalışacağını da dinamik değişkenler kullanarak göz önüne alıyor.

2.4. Program Çalıştırma Seçenekleri

HCS12 EmumiS bir programı 3 farklı şekilde çalıştırma imkânı vermektedir:

1. *Tüm programın benzetimi ve gerçek zamanlı geliştirme kartı öykünmesi*: Bu yöntemde programın gerçekte tüm çalışma süresince olacak olaylar ve zamanları yani senaryo hesaplanır ve daha sonra bu senaryo görsel olarak geliştirme kartı üzerinde kullanıcıya sunulur.
2. *Assembly seviyesinde adım adım çalıştırma*: Programın komutları teker teker çalıştırılarak ve her defasında hem geliştirme kartı hem de HCS12 bellek ve yazmaçlarında gerçekleşen tüm değişimler gözlemlenerek programdaki hatanın tespiti yapılabilir.
3. *Duraksatma noktasına kadar çalıştırma*: Adım adım çalıştırma gecikme gibi binlerce tekrarlı döngülerde kullanıcıyı yavaşlatır. Böyle durumlarda hata ayıklamayı mümkün kılmak için programın durması istenen komutlara duraksatma noktaları konularak o komutlara kadar çalıştırma gerçekleştirilir.

2.5. Gerçek Zamanlı Öykünme Problemi ve Çözümü

Bölüm 2.4'teki 1 numaralı çalıştırma seçeneğinde, kullanıcı tüm programın geliştirme kartı üzerinde çalışmasını görsel olarak EmumiS üzerinden görme imkanına sahip olmaktadır. Programın tamamının gerçek zamanlı geliştirme kartı öykünmesini çalıştırmada kritik olan, sanal geliştirme kartı üzerindeki bileşenlerin sunumlarını doğru zamanda gerçekleştirmenin zorluğudur. Oysa doğru bir öykünme için, geliştirme kartında doğru bileşenin, doğru zamanda, doğru işi yapması gerekmektedir. Çünkü zaman benzetimi yanlış olan görsel veya işitsel bir sunum geliştirici için faydalı olmaktan çok anlamsız ve dahası aldatıcı olacaktır.

HCS12 EmumiS, hem komut ayrıştırma ve metot çağırma hem de benzetim ve öykünme aynı anda yaptığında benzetim performansı olarak fiziksel HCS12'ye nazaran daha yavaş bir sunum performansı sunmaktadır ki bu gerçek zamanlı öykünmeyi olanaksızlaştırır. Bu sebeple; EmumiS önce mikrodenetleyici komutlarını ayrıştırıp tüm öykünme senaryosunu belirlediği *değerlendirme* aşamasıyla işe başlar. Bu aşamada kullanıcıya bir şey gösterilmez. İkinci aşamada ise kullanıcıya gerçek zamanlı HCS12 benzetimini ve geliştirme kartı üzerinde değerlendirilmiş olan senaryonun öykünmesini sunar. Kullanıcının, tüm programın geliştirme kartı üzerinde çalışmasını izlediği bu ikinci aşama da *sunum* aşamasıdır. Bu strateji sayesinde geliştirme kartı öykünmesinde görselliği hiç etkilemeyen sadece milisaniyeler düzeyinde sapmalar olup bunun da sebebi üzerinde çalıştığımız bilgisayarın işletim sisteminin gerçek zamanlı olmamasıdır.

Gerçek zamanlı öykünmeyi zora sokan başka bir problem de 7-segment LED gibi bileşenlerin aşırı hızlı yakılıp söndürülmesi durumudur. LED'lerin yanma durumu 1 µs'den daha kısa süre aralıklarıyla değişebilmektedir. Bu kadar küçük süreçlerdeki değişimleri sunum aşamasında bu hızda yapmak müthiş bir yük getirmektedir. Bu da sunum zaman benzetimini olumsuz etkileyip kötü sonuçlar elde etmemize neden olmaktadır.

Böyle bir performans problemi olan 7-Segment ve LED bileşenlerinin muhatabının insan gözü olması ve insanın görüş hızının sınırlı olması, bize bu problemin çözümünde yol gösterdi. İnsan gözü olayları 18-25 kare/saniye ile algılayabilmektedir. İnsan gözü, 20 milisaniyeden daha kısa sürelerde yakılan farklı LED'lerin hepsini birden yanıyor olarak algılayacağından, sunumda yüksek hızda yakıp söndürmek yerine hepsini yanık olarak gösterdik. Bu sayede EmumiS hem insanın gerçekte gözlemleyeceğine tamamen uygun bir geliştirme kartı öykünmesi sunmakta hem de çok hızlı LED durumu değişikliklerinden ötürü doğan performans sıkıntısını gidermektedir.

2.6. Girdi Problemi ve Çözümü

HCS12 EmumiS'in önemli problemlerinden birisi geliştirme kartında yer alan dip-anahtar girdilerini kullanıcıdan almadır. Girdi vermesi gerektiğinde bunu kullanıcı ne zaman ve nasıl yapacak? Eğer girdileri değerlendirme aşamasından önce girerse bu defa kullanıcı geliştirme kartı öykünmesini hiç izleyemeden girdileri neye göre belirleyecek? Ayrıca bu girdileri geleceğe dönük ne zaman gireceğini nasıl belirleyecek? Eğer girdileri daha sonra sunum aşamasında

girerse, değerlendirme aşamasında kullanıcının gireceği girdiler ortada yokken değerlendirme neye göre olacak?

Bu problemlere çözüm olarak değerlendirme ve sunumu girdi değerine bakılan zamana kadar yapma ve girdi gerektiğinde senaryo sunumunu kesme ve kalan yerden değerlendirme ve sunum yapma stratejisini geliştirdik. Buna göre, senaryosu değerlendirilen programın kullanıcıdan bir girdi okumak istediği tespit edildiğinde, değerlendirme aşaması kesilerek o ana kadar değerlendirilen senaryonun sunumu gerçekleştirilmektedir. Böylece kullanıcı aynen gerçekte olacağı gibi girdisini girmeden önce geliştirme kartı üzerindeki tüm değişiklikleri izleyebilir ve girdisini ona göre belirleyebilir. Kullanıcı tarafından girdi yapılmasıyla birlikte senaryo değerlendirme aşamasına tekrar dönülür ve bu kez senaryo değerlendirilirken kullanıcının girdiği girdiler dikkate alınır. Program içerisinde kullanıcıdan kaç kere girdi alınacaksa bu aşama geçişleri döngü halinde devam eder. Bu strateji sayesinde kullanıcı hem girdiden önceki geliştirme kartındaki değişiklikleri görsel olarak izleme, hem de girdisini verdikten sonra onun öykünmeye etkisini yine izleme imkânı bulabilmektedir. Gerçek hayattan farklı olan tek durum ise kullanıcının her girdiden sonra gerçekte beklemediği değerlendirme sürecinin bitmesini bekleme zorunluluğudur. Girdi isteğinin tespiti sonraki bölümde anlatılmaktadır.

2.7. EmumiS Abonelik Sistemi

Gerek geliştirme aracı bileşenlerinin çalışma, gerekse de HCS12 zaman ve SCI modüllerinin çalışma prensipleri belli bellek adreslerinin değerlerini gözlemleme ve değişikliklere göre davranmayı gerektirir. Örneğin, Dragon12 Geliştirme Aracı'nda sekiz adet LED bileşeni HCS12'nin PORT B bacağına ve 0x01 bellek adresine bağlı olup, LED'ler bu adresteki verinin değerine göre yanıp sönmektedir. Geliştirme kartı öykünmesi yapılırken bir yandan sürekli her bir belleğin değişikliğini takip etmek çok ağır bir yüke neden olacaktır.

Bu sebeple; her bir geliştirme kartı çıktı bileşeni, EmumiS'in değerlendirme aşamasında, kendi bağlı bulunduğu bellek adresinin içeriğini takip edip gerçekleşen değişiklikleri ve bunların zamanını kaydetmektedir. Gözlemci Tasarım Örüntüsünden (Observer Design Pattern) [16] esinlenerek geliştirdiğimiz bu abonelik sisteminde çıktı bileşenleri (LED, LCD…) veya HCS12 SCI ve zaman modülleri gibi modüller kendileriyle ilgili bellek adreslerine abone olarak "gerektiği zaman bize haber ver" ilişkisini kurarlar. EmumiS, bu abonelik sistemini kullanarak aynı zamanda dip-anahtar girdisine ihtiyaç duyulduğunu anlayabiliyor. Ancak bu kez "erişilmek istendiğinde bize haber ver" ilişkisi kullanılır. Bu sayede girdi istendiği anda tespit edilir ve Bölüm 2.6'da anlatılan çözüm devreye girebilir.

2.8. Zaman Modülü Benzetimi

HCS12 zaman modülü 16-bit'lik çevrimleri sayan sayaca sahiptir. Sayaç gerekli ayarların yapılması ile 0x0000 değerinden saymaya başlayıp hızına göre belli bir süreçte 0xFFFF olan maksimum değere ulaşıp taşmakta ve başa dönmektedir. Zaman modülünde yer alan ön-çarpıcı (prescalar) sayesinde sayacın sayma hızı ve taşma süresi ve işlemci frekansı ayarlanabilmektedir. Geçen zamanın belirlenmesi, senaryonun doğru zamanlamayla gösterilmesi için zamanın kaydının EmumiS tarafından sürekli tutulması gerekir. Sayacın etkinleştirilmesi, ön-çarpıcıda herhangi bir değişiklik olması gibi zamanlama ile ilgili tüm değişiklikler Bölüm 2.7'de anlatılan abonelik sistemi ile ilgili bellek adreslerine abone olunarak takip edilmektedir. Ayrıca, eğer sayaç aktif ise her bir komut çalıştırılmasının ardından komutun harcadığı çevrim sayısına göre sayaç değeri güncellenmektedir. Bu yapıyla HCS12 zaman sayacı benzetimi hatasız bir şekilde gerçekleştirilmiştir.

HCS12 zaman modülünün başka bir özelliği de Çıktı Kıyas (Output Compare) sistemidir. Çıktı Kıyas sistemi PORT T'nin herhangi bir bacağında istenen sayısal (mantıksal) bir değişikliğin (mantıksal 1'den 0'a ya da 0'dan 1'e değişim) istenilen anda kendiliğinden gerçekleşmesini sağlar. Bu şekilde istenen frekansta kare dalga da üretmeye yarayan çıktı kıyas sistemi için TIOS, TCx, TCTL1 ve TCTL2 gibi Çıktı Kıyas bellek adreslerinin hepsi EmumiS tarafından önceki bölümde anlatılan abonelik sistemi ile takibe alınmıştır. Onlarda bulunan değerlere göre PORT T'nin hangi bacağında, hangi değişikliğin, ne zaman yapılması gerektiği değerlendirme aşamasında belirlenip, sunum aşamasında kullanıcıya doğru Çıktı Kıyas senaryosu sunulmaktadır.

EmumiS'in benzetebildiği diğer bir zaman modülü özelliği Girdi Yakalama (Input Capture) sistemidir. Girdi Yakalama özelliği sayesinde PORT T'nin önceden belirlenmiş herhangi bir bacağında meydana gelmesi beklenen belirli bir sayısal değişikliğin gerçekleşmesi durumunda o andaki sayaç değeri kendiliğinden ilgili bellek adresine yazılır. EmumiS, Girdi Yakalama benzetimini yapmak için TIOS, TCx, TCTL3 ve TCTL4 gibi Girdi Yakalama sistemi tarafından kullanılan bellek adreslerini yine abonelik sistemi ile takibe almıştır. Ayrıca, eğer Girdi Yakalama etkinleştirilmişse EmumiS PORT T'yi de aboneliğe alır. Bu abonelikler sayesinde portun istenen bacağında, istenen mantıksal değişikliğin oluştuğunu tespit eder ve o anki sayaç değerini ilgili bellek adresine kaydeder.

2.9. SCI (UART) Modülü Benzetimi

SCI modülü HCS12'nin UART protokolü ile seri olarak başka bir modül ile iletişim kurmasını sağlar. Normalde bilgisayarda yer alan bir mikrodenetleyici öykünücüsünde bu modülün benzetiminin yapılması çok zordur, çünkü bilgisayar yazılımından ibaret olan öykünücülere UART kullanan başka bir cihazı takmanız mümkün değildir ve iki yönlü haberleşmeye öykünmeniz de anlamlı olmamaktadır.

Ancak SCI modülü gibi iletişim modülleri gömülü sistem geliştirme ve mikrodenetleyici iletişim protokolleri eğitiminde hayati öneme sahiptir. Bu sebeple EmumiS çalıştığı bilgisayarın USB portunu bir sanal COM portu gibi kullanarak dışarıdan USB üzerinden bağlanabilen cihazlarla haberleşebilecek şekilde tasarlandı. Bu sayede HCS12 EmumiS ile kullanıcılar, SCI modülünün çalışmasının öykünmesini de yapabilmektedir. Ayrıca, EmumiS arayüzünde bir bileşen olarak yer alan terminali sanal olarak geliştirme kartına bağlayıp bu terminal üzerinden HCS12 SCI modülü ile irtibat kurabilmektedirler. EmumiS, SCI modülünün BAUD hızı, eşlik (parity) biti gibi tüm ayarlamalarını desteklemektedir. Bu sayede EmumiS, farklı özellikleri destekleyen fiziksel cihazlarla USB üzerinden sanal olarak irtibata geçebilir.

2.10. Kesmelerin (Interrupt) Benzetimi

Kesmeler de genelde öykünücülerde ihmal edilen ancak özellikle eğitim açısından önemli sistemlerden birisidir. HCS12'de her bir kesme için bellekte 0xFF80 ile 0xFFFF arasında bulunan vektör tablosunda bir adres ayrılmıştır. Bir kesme oluştuğu zaman kesme vektör tablosunda bulunan ilgili adres Program Sayıcıya (Program Counter) yüklenir. Böylece HCS12'nin rutin çalışması kesilerek kesme fonksiyonunun çalışması sağlanır. Kesme işlemi gerçekleştiği anda tüm yazmaç değerleri yığında (stack) yedeklenir ve kesmenin ardından yedeklenen bu değerler tekrar yüklenilerek kesme gelmeden önceki hale dönülür. Bölüm 2.1'de desteklendiği açıklanan tüm kesmeler, EmumiS'te aynen bu şekilde çalışacak şekilde tasarlanmıştır.

3. Testler ve Sonuç

HCS12 EmumiS, HCS12 komut kümesinin %94'ünü tanımaktadır. Yazdığımız ve derlediğimiz programlarda kalan %6'lık kısımda kalan komutlarla hiç karşılaşmadık. LCD, LED'ler, 7-segment LED, osiloskop, terminal (SCI), vızlayıcı, dip-anahtarlar gibi tüm geliştirme kartı bileşenlerini kullanan dokuz farklı test programı yazdık. Genel Demo tüm bileşenleri birden kullanan bir test programıydı. Testler sonucunda fonksiyonel olarak HCS12 EmumiS doğru sonuçlar verdi, yani diğer deyişle geliştirme kartı öykünmesi ve HCS12 benzetiminde bize gerçekte, fiziksel Dragon12 kartı üzerinde de gözlemlediğimiz doğru senaryoları gösterdi.

Çizelge 1'de test programları ve EmumiS performansını görebilirsiniz. Bu programlar kurulumla [14] beraber gelmekte, ayrıca uygulanan bazı testlere ait videolara [17]'den ulaşılabilmektedir. Testler, Intel i5-430M (2,53 Ghz), 3 GB RAM donanımı ve Windows 8.1 (32 bit) işletim sistemi ile gerçekleştirilmiştir. Test programlarının her biri fiziksel olarak Dragon 12 geliştirme kartı üzerinde çalıştırıldığında geçen süreyi gerçekte geçen süre sütununda gösterdik. HCS12 EmumiS Değerlendirme aşamasında geçen ve Sunum aşamasında geçen süreleri ise sırasıyla diğer sütunlarda gösterdik. Buna göre, HCS12 EmumiS'in sunum süreleri gerçekte geçen sürelerden insanın algılayabileceğinden çok daha küçük sapmalar göstermiştir. Bununla beraber örneğin Genel Demo programında kullanıcı, senaryoyu izlemeden önce EmumiS'in değerlendirme aşamasıyla senaryoyu tespit etmesi için 46 saniye beklemek zorundadır.

Test Edilen Program	Gerçekte Geçen Süre (sn)	EmumiS Değerlen. Süresi (sn)	EmumiS Sunum Süresi (sn)
Genel Demo	11,8	46	11,806
Zaman Taşması	10	20,4	9,986
Vızlayıcı testi	7,6	26,9	7,21
Hesap Makinesi	14,9	29,3	14,982
Saat	10	18,5	9,983
LCD testi	4,3	18,3	4,099
LED testi	10	29,5	9
SCI testi	10	15	9,975
7-Seg. Sayaç	10	17,2	9,983

Çizelge 1: Değerlendirme ve Sunum Aşaması Performansı

Performans bir programda komut başına düşen çevrim sayısı (CPI) ile de doğru orantılıdır ve bunun artması birim sürede daha az komutun çalıştırılması yani daha az fonksiyon çağırma işleminin gerçekleşmesi anlamına geldiğinden performansı olumlu etkilemektedir.

Sonuç olarak, Bölüm 2.1'de listelenen tüm özelliklere sahip olan HCS12 EmumiS, sadece HCS12 değil, tüm mikrodenetleyicileri benzetirken ve geliştirme kartlarına öykünürken karşılaşılacak birçok probleme ışık tutmuş ve makul çözümler getirmiştir. Bu ve bunun gibi sanal mikrodenetleyici geliştirme araçları; kendini bu konuda geliştirmek isteyen herkese, geliştirme kartı satın alma ve temin etme gibi problemlerle uğraşmadan doğrudan ve ücretsiz şekilde mikrodenetleyicileri öğrenme ve kendini geliştirme imkânı sağlayacaktır.

4. Kaynaklar

[1] Huang, H., *HCS12/9S12 An Introduction to Software and Hardware Interfacing*, Delmar Cengage Learning, USA, 2006

[2] *S12CPUV2 Reference Manual*, Freescale, USA, 2006

[3] Sumey, J., CodeWarrior Familiarization & Project Setup, University of Pennsylvania, USA, 2011, *http://www.aet.calu.edu/~jsumey/CET360/cwintro/cwintro.html*

[4] Koopman, P., ECE 348 Embedded System Engineering Syllabus, Carnegie Mellon University,USA, 2014, *http://users.ece.cmu.edu/~koopman/ece348/*

[5] Valvano, J. W., Personal Educator Page, University of Texas, USA, 2014, *http://users.ece.utexas.edu/~valvano/*

[6] Ume, C., Mechatronics Laboratory Syllabus, Georgia Institute of Technology, USA, 2014, *http://ume.gatech.edu/mechatronics_lab/*

[7] Mazidi, M. ve Causey, D., *HCS12 Microcontrollers and Embedded Systems*, Prentice Hall, USA, 2008

[8] Pack, D., *68HC12 Microcontroller : Theory and Applications*, Prentice Hall, USA, 2002

[9] Valvano, J., *Introduction to Embedded Systems: Interfacing to the Freescale 9S12*, Cengage Learning, UK, 2009

[10] Dragon12-Plus,EVBplus, *http://www.evbplus.com/9s12/9s12_hcs12.html*

[11] LFEBS12UB, Freescale, *http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=LFEBS12UB*

[12] CodeWarrior Development Tools, Freescale, *http://www.freescale.com/webapp/sps/site/homepage.jsp?code=CW_HOME*

[13] SIMHCS12, Tom Almy, *http://www.hcs12text.com/freesim.html*

[14] HCS12 EmumiS, *http://bilmuh.gyte.edu.tr/~bayrakci/EmumiS.html*

[15] OxyPlot, MIT, *https://github.com/oxyplot*

[16] E. Gamma, R. Helm, R. Johnson ve J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995

[17] *http://www.youtube.com/channel/UCP8jvWsoa8HC1sXlwfPKnHw/videos*