VSCPU ile Eğitsel Açıdan İşlemci Komut Kümesinin Öneminin Ele Alınması

Abdullah Yıldız*, Tugay Emre Yucedag*, H. Fatih Ugurdag†, and Sezer Gören*
*Dept. of Computer Engineering, Yeditepe University, Istanbul, Turkey
Email: ayildiz@cse.yeditepe.edu.tr, eyucedag@cse.yeditepe.edu.tr, sgoren@cse.yeditepe.edu.tr
†Dept. of Electrical and Electronics Engineering, Özyeğin University, Istanbul, Turkey
Email: fatih.ugurdag@ozyegin.edu.tr

***Özetçe* —Bu bildiri, işlemci tasarımında önemli bir unsur olan işlemci komut kümesinin eğitsel açıdan uygulamalı olarak incelenmesi gerektiğini vurgulamak amacıyla ortaya koyduğumuz VSCPU (başka bir deyişle VerySimpleCPU) işlemci mimarisini ve onun toolchain'ini tanıtmaktadır. Bildiride VSCPU'nun eğitsel önemini ve derslerde öğrenciler tarafından ilgili dersler kapsamında uygulamalı olarak nasıl kullanıldığını da kısaca anlatmaktayız. Son olarak, öğrencilere VSCPU'nun da evrensel yani bir Turing-complete işlemci olduğunu göstermek için PIC mikrodenetleyicileri için yazılmış programları VSCPU üzerinde de çalıştırabilmemizi sağlayan cross-assembler betiği açıklanmaktadır.**

Anahtar Kelimeler—işlemci, işlemci komut kümesi, işlemci mimarisi, FPGA, mikrodenetleyici.

I. GİRİŞ

İşlemci, bir bilgisayardaki esas unsurdur. Günümüzde arama motoru, sinirsel ağlar, görüntü işleme gibi belirli bir alana özgü uygulamaların önem kazanmasıyla birlikte ortaya çıkan ve bu uygulamaların performans ve güç ölçütlerini iyileştirmek maksadıyla tasarlanan özel amaçlı işlemciler ya da yardımcı işlemciler (coprocessor) de tıpkı genel amaçlı bir işlemci gibi bir komut kümesine sahiptirler. Dolayısıyla, komut kümesi bir işlemcinin en önemli ayırt edici özelliğidir, diyebiliriz.

Bir işlemcinin komut kümesinin neden belirli komutlardan oluştuğunun eğitsel açıdan incelenmeye değer olduğunu düşünüyoruz. Bu amaçla, ilgili mühendislik derslerinin müfredatında bunun yer etmesi, öğrencilere komut kümesinin işlemci mimarisindeki yeri açısından farklı bir bakış açısı kazandırabilir.

Buna ek olarak, öğrencilerin sayısal tasarım ve bilgisayar mimarisi ilgili derslerde karşılaştıkları temel sorunlardan biri de bir tümdevrenin tasarım girdi ile tasarımına başlanması aşamasından silikon yonga haline getirildiği üretim aşamasına kadar olan süreci deneyimlemelerinin mümkün olmamasıdır. Bu sebepten ötürü, günümüzde yeniden yapılandırılabilir kapı dizileri yani FPGA tümdevrelerinin ve bu tümdevrelerin EDA araçlarının erişilebilirliğinin artması sayesinde bugün öğrenciler bir nebze de olsa bu sürece ilişkin bir deneyim kazanmaktadırlar diyebiliriz.

Biz bu bildiride, işlemci tasarımını eğitsel açıdan ele aldık ve öğrencilerin uygulamalı olarak ve kendi katkılarını ortaya koyabileceği bir işlemci mimarisi sunmak istedik. Bu amaçla, VSCPU (başka bir deyişle VerySimpleCPU) ve toolchain'ini geliştirdik. Böylelikle öğrenciler, FPGA ve ilgili EDA araçlarını etkin olarak kullanarak işlemci tasarım ve geliştirme sürecini deneyimleyebilmektedirler.

Bu bildirinin organizasyonu şu şekildedir: Bölüm II'de VSCPU detaylı olarak tanıtılmakta ve toolchain'i kısaca açıklanmaktadır. Ayrıca VSCPU'yu eğitsel olarak nasıl kullandığımız ile ilgili kısaca bilgi verilmektedir. Bölüm III'de, VSCPU'nun da evrensel bir işlemci olduğunu göstermek için VSCPU ve komut kümesi akademide ve endüstride yaygın olarak kullanılan PIC16F mikrodenetleyicilerinin işlemcisi ve komut kümesiyle karşılaştırılmakta ve PIC için yazılmış programları VSCPU programlarına çeviren cross-assembler betiğimiz hakkında bilgi verilmektedir. Bölüm IV'da çalışmalarımızda elde ettiğimiz çıkarımlar tartışılmakta ve gelecek çalışmalarımız hakkında bilgi verilmektedir.

II. VSCPU

VSCPU [1], genel olarak basit ve özelleştirilebilir olmasının yanında bilgisayar mühendisliği müfredatında yer alan bilgisayar mimarisi veya bilgisayar organizasyonu derslerinde kullanılmak üzere tasarlanmış bir soft işlemcidir. 2012 yılında tasarlandığından bu yana ilgili dersler kapsamında 2 farklı üniversitede lisans seviyesindeki öğrencilere işlemci tasarımını uygulamalı olarak öğretmek amacıyla etkin olarak kullanılmaktadır. VSCPU işlemcisine ait komut kümesi simülatörü, assembler ve C derleyicisinden oluşan bir toolchain de mevcuttur. Bu bağlamda dikkate çekmeye değer başka bir nokta da VSCPU'nun sahip olduğu yazılım ekosistemiyle birlikte ortaya çıkmış ilk **yerli** işlemci olmasıdır.

VSCPU'nun sade ve basit olarak tasarlanmasının sebeplerinden birincisi, öğrencilerin bir işlemcinin komut kümesi ile o işlemcinin çalıştıracağı programların içerdiği komut sayısı arasındaki ilişkiyi (burada dolaylı olarak işlemcinin performansını da dahil edebiliriz) anlamalarını sağlamaktır. Şöyle ki; bir işlemci sadece **NAND** komutunu içerebilir (örn. NAND 10 20 gibi), ancak böyle bir işlemcide çalıştırılabilecek programların ihtiyaç duyduğu aritmetik, mantık, kopyala, dallan vb. işlemlerin her biri için sadece NAND komutunu kullanarak bu işlemlere karşılık düşen komut bloklarını *"sentezlemek"*, programı yazmak açısından oldukça zahmetli ve zaman alıcı bir sürece dönüşecektir. Öte yandan, derleyicilerin performansa dayalı yapacağı iyileştirmelere olanak tanımak için onlarca komut içeren ARM ya da MIPS ve yüzlerce komut içeren x86 mimarilerini ele aldığımızda da bu kez komut çeşitliliğinin yüksek olması öğrencilerin bu komutları öğrenme ve

bu komutlarla program yazma pratiğini elde etme sürecini uzatacaktır. Burada program yazmaktan kastımız işlemcinin komut kümesini kullanarak assembly seviyesinde program yazmaktır. Bir işlemcinin çalışma mekanizmasını anlayabilmek için bunun mutlak gerekli olduğu açıktır. Bu düşünceyi göz önüne alarak, VSCPU komut kümesi içinde yer alan komutlar, bir işlemcinin sahip olması gereken temel ve asgari komut kümesini oluşturacak, aynı zamanda da öğrencilerin bu komut kümesi üzerinde çeşitli problemlere yönelik değişiklik ve iyileştirmeler yapabilmelerini sağlamak üzere ortaya konmuştur.

İkinci sebep olarak da, işlemcinin sadece performansını (dolaylı olarak enerji verimliliğini) arttırmak için günümüz işlemcilerinde sıklıkla kullanılan önbellek (cache), dallanma tahmin (branch prediction), düzensiz yürütme (out-of-order execution) gibi mekanizmaların işlemcinin asıl unsuru olmamasıdır. Bu mekanizmaların içerisinde sadece boruhatlama (pipelining) mekanizmasını VSCPU'ya dahil etmemizin sebebi, boruhatlamanın sayısal tasarımda çok sıklıkla tercih edilen bir yöntem olması ve öğrencilere bu yöntemin bir işlemci için faydasını göstermektir.

A. Mimari

VSCPU 32-bit mimariye sahip bir işlemci çekirdeğine sahiptir ve her bir operandın sadece bellekte olduğu *bellek-bellek* [2] komut kümesi mimarisini temel alır. Bilgisayar mimarisi açısından ise hem *von Neumann* hem de *Harvard* bilgisayar mimarilerinden birisini kullanarak gerçekleştirilebilir. VSCPU'nun iki farklı komut kümesi mimarisi bulunmaktadır: ISAv1 işaretsiz tamsayı aritmetik işlemleri, ISAv2 ise işaretli tamsayı ile tek duyarlıklı kayan nokta aritmetik işlemlerini destekler. Her iki komut kümesi de aritmetik/mantık, taşıma ve program kontrol işlemlerini destekleyen toplam 16 komuttan oluşur ve her bir komut kelimesi 32-bit'ten meydana gelir. Tablo I VSCPU komut kelimesinin içeriğini göstermektedir.

TABLO I: VSCPU komut kelimesi

	Instruction Word		
bit position	[31:28]	[27:14]	[13:0]
field name	opcode	operand A	operand B
bit width	4-bit	14-bit	14-bit

VSCPU'nun temel özellikleri şu şekilde sıralanabilir:

- 32-bit bellek+bellek komut kümesi mimarisi
  - ISAv1 - işaretsiz tamsayı aritmetiği desteği
  - ISAv2 - işaretli tamsayı + tek duyarlıklı kayan nokta aritmetiği desteği
- von Neumann/Harvard bilgisayar mimarisi
- Yazmaçlar
  - 14-bit Program Sayacı Yazmacı (PC)
  - 32-bit Komut Kelimesi Yazmacı (IW)
  - 32-bit Akümülatör (ACC ya da R1)
- Boruhatlı/Boruhatsız tasarım
- Kesme işlevi desteği
- Bellek haritalı G/Ç desteği

Tablo I'de görüldüğü üzere her bir komut kelimesi aynı biçimde bellekte yer alır ve üç farklı alandan oluşur: en soldaki 4-bit'lik *opcode* alanı komut işlem kodunu, 14-bit'lik *operand A* ve *operand B* alanları ise komut kelimesi opcode alanındaki değere göre bellekteki bir değerin bellek adresini ya da dolaysız (immediate) veriyi temsil eder. Tablo II'de VSCPU ISAv1'e ait komut kümesi her bir komutun tanımı ve işleviyle birlikte verilmiştir. ISAv1 ile ISAv2 arasındaki temel ayrım daha önce de belirtildiği gibi desteklenen aritmetik işlemlerin farklılığıdır ve bu sebepten bu bildiride ISAv2 komut kümesinin verilmesine gerek duyulmamıştır.

Tablo II'de gösterildiği üzere VSCPU 3 farklı komut türüne sahiptir: aritmetik & mantık, taşıma ve program kontrol komutları. Kırmızı renkle ve kalın harflerle vurgulanan komutlar yukarıda bahsedildiği gibi VSCPU komut kümesinin temel (fundamental) ve asgari (minimal) nitelikleri taşıması açısından uygun gördüğümüz komutlardır. Diğerleri bu komutlardan sentezlenebilir fakat kolaylık olması ve ortaya konulan programlama çabasını arttırmamak için komut kümesine dahil edilmişlerdir.

Bunun yanında, her bir komut VSCPU'nun bellek+bellek mimarisini kullanması dolayısıyla işini tamamlayabilmek için birden fazla saat çevrimine ihtiyaç duyar. VSCPU komut çevrimine ilişkin diyagram Şekil 1'de gösterilmiştir.

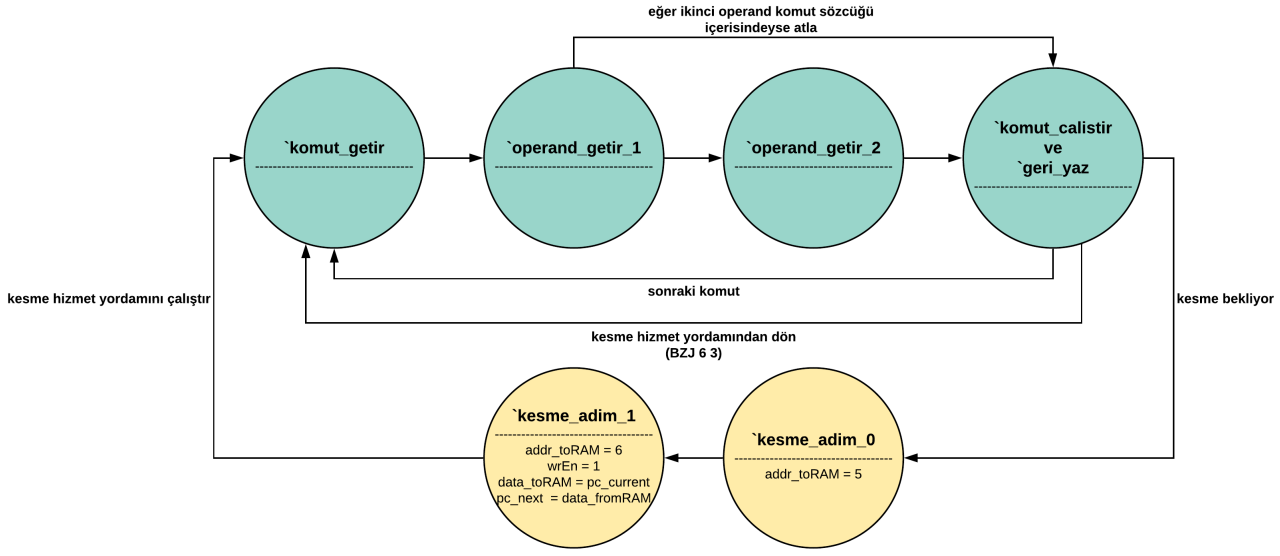
Şekil 1'de gösterildiği üzere, VSCPU kesme işlevini de desteklemektedir. Bunun için bellekte beşinci ve altıncı adresler sadece kesme mekanizmasını işletmek üzere özel adresler olarak ayrılmıştır. Beşinci bellek adresi, kesme servis rutininin (ISR)'in başlangıç komut adresini ve altıncı bellek adresi de kesme servis rutini tamamlandıktan sonra çalışacak bir sonraki komutun adresini gösterir. Bu mekanizma, VSCPU'nun komut çevrimini işleten durum makinesinin içerisinde gerçeklenmektedir. Buna göre, komut çevrimi esnasında meydana gelebilecek herhangi bir kesme olayı VSCPU içerisinde servis edilmek üzere o anki komut çevriminin tamamlanması beklenir. Komut çevrimi tamamlanırken, ISR'dan dönüldükten sonra çalışacak komutun adresi altıncı bellek adresine yazılır ve kesmenin servis edilebilmesi için beşinci bellek adresinde bulunan komut adresine yönlendirme gerçekleştirilerek ilgili ISR çalıştırılır. VSCPU'da ISR'dan dönme ile ilgili bir komut olmadığı için, bu amaçla **BZJ 6 3** komutu kesmeden dönmek için her ISR sonunda çalıştırılır (Üçüncü bellek adresinde sabit sıfır değeri bulunduğu için işlemci bir sonraki çalıştıracağı komut adresini altıncı bellek adresinden alacaktır).

B. Tasarım

VSCPU'nun donanımsal tasarımı Şekil 1'e göre farklı yöntemlerle gerçekleştirilebilir. Bizim tercih ettiğimiz yöntem, sayısal tasarım derslerinde sıklıkla kullanılan bir yöntem olan sonlu durum makinesi (finite state machine) modeli oldu. Bunu donanımsal olarak gerçekleştirebilmek için de elektronik tasarım otomasyonu (EDA) araçlarında en çok tercih edilen tasarım girdi (design entry) türlerinden biri olan Verilog donanım tanımlama dilini (HDL) tercih ettik. VSCPU'nun Verilog HDL ile kara kutu (black box) modeli Liste 1'de verilmiştir (burada VSCPU tasarımının tamamını vermeye gerek duymadık). Görüldüğü üzere saat ve reset dışında bellekle haberleşebilmek için 4 portu bulunmakta ve 14-bit'lik *addr_to_mem* portuyla 64KB'lık bellek alanına erişebilmektedir.

TABLO II: VSCPU ISAv1 komut kümesi

Mnemonic	**Tanım**	**İşlev**
Aritmetik & Mantık Komutları		
ADD A B	işaretsiz tamsayı topla	`*A = *A + *B`
ADDi A B	işaretsiz tamsayı topla, dolaysız	`*A = *A + B`
NAND A B	bit bit NAND	`*A = ∼(*A & *B)`
NANDi A B	bit bit NAND, dolaysız	`*A = ∼(*A & B)`
SRL A B	sağa/sola kaydır	`*A = (*B < 32) ? (*A » *B) : (*A « (*B−32))`
SRLi A B	sağa/sola kaydır, dolaysız	`*A = (B < 32) ? (*A » B) : (*A « (B−32))`
LT A B	karşılaştır	`*A = *A < *B`
LTi A B	karşılaştır, dolaysız	`*A = *A < B`
MUL A B	işaretsiz tamsayı çarp	`*A = *A x *B`
MULi A B	işaretsiz tamsayı çarp, dolaysız	`*A = *A x B`
Taşıma Komutları		
CP A B	kopyala	`*A = *B`
CPi A B	kopyala, dolaysız	`*A = B`
CPI A B	kopyala, dolaylı	`*A = **B`
CPIr A B	ters kopyala, dolaylı	`**A = *B`
Program Kontrol Komutları		
BZJ A B	sıfırsa dallan	`PC = (*B == 0) ? *A : (PC + 1)`
JMP A B	koşulsuz dallan	`PC = *A + B`



Şekil 1: VSCPU komut akışı durum diyagramı [1]

```
module VerySimpleCPU(

  // saat portu
  input wire [0:0] clk,
  // reset portu
  input wire [0:0] rst,

  // 14-bit bellek adresleme portu
  output reg  [13:0] addr_to_mem,
  // 32-bit bellek veri okuma portu
  input  wire [31:0] data_from_mem,
  // 32-bit bellek veri yazma portu
  output reg  [31:0] data_to_mem,
  // 1-bit bellek veri yazma gecerli portu
  output reg  [0:0]  wr_en

);

endmodule
```

Liste 1: VSCPU'nun Verilog HDL ile tanımlanmış kara kutu modeli

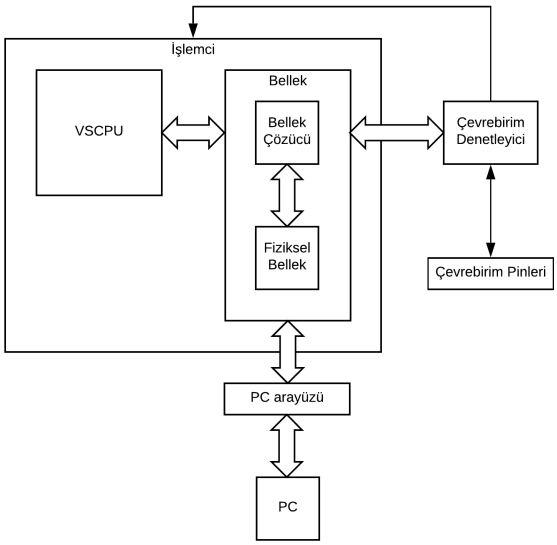
C. Doğrulama ve Gerçekleme

VSCPU'nun saat bazında (cycle-accurate) benzetimini yapmak ve çalıştığını doğrulamak için *Xilinx ISE* yazılım paketi içinde yer alan *ISim* benzetim aracını kullanmaktayız. VSCPU'yu FPGA tümdevresine yönelik sentezlemek ve gerçeklemek için yine *Xilinx ISE* yazılım paketi içinde yer alan sentezleyici (*xst*), eşleştirme (*map*), yerleştirme-yol atama (*par*) ve bitstream oluşturma (*bitgen*) araçlarını kullanmaktayız.

VSCPU'yu öğrencilerin benzetim ile doğrulaması amacıyla işlemci üzerinde çalışacak program için gerekli büyüklükte bir bellek kullanmak yeterli olacaktır (yaklaşık 1KB), ancak bir işlemcinin işleme (processing) dışındaki bir diğer esas görevi de giriş-çıkış (G/Ç) aygıtlarıyla etkileşimde bulunabilmesidir. Bu sebeple, VSCPU'yu basit bir mikrodenetleyici oluşturmak için kullanabiliriz. Şekil 2'de bu mikrodenetleyici tasarımına ilişkin blok diyagramında gösterildiği üzere, işlemci içerisinde VSCPU ile bellek ünitesi yer almaktadır. Bellek, çekirdekten (VSCPU) erişilen adreslerin fiziksel bellek (Physical Memory) ya da çevrebirimlere karşılık düşüp düşmediğini kontrol eder ve buna göre ilgili veri akışı sağlanır. Çevrebirim denetleyicisi (Peripheral Controller), çevrebirimlerin kontrol devrelerini içeren kısımdır ve asıl çevrebirim ünitelerine çevrebirim pinleri (Peripheral Pins) üzerinden erişim sağlar. Ayrıca, PC (Host) ile işlemci (CPU) arasında program yüklemesi (download) ve hata ayıklama (debug) işlemleri için bir arayüz de tasarlanmıştır. Şimdiye kadar öğrencilerimiz çeşitli çevrebirimlere erişebilen uygulamalar üzerinde çalışmışlardır. Bu çevrebirimler şu şekildedir:

- Genel Amaçlı Giriş-Çıkış ya da GPIO: örn. LED, push button, switch vb.
- Darbe Genişlik Modülasyonu (PWM): örn. DC motor
- Tümdevreler Arası Arayüz (I2C): örn. ivmeölçer, optik sensör
- Seri Çevrebirim Arayüzü (SPI): örn. ivmeölçer
- Evrensel Asenkron Alıcı-Verici (UART): örn. PC terminal
- Video Grafik Dizisi (VGA)

Bu çevrebirimlerden UART ve VGA hariç diğerlerinin çevrebirim denetleyicilerini Verilog HDL ile tasarladık, doğruladık ve çeşitli G/Ç aygıtları üzerinde kullanmaktayız. UART ve VGA çevrebirimlerinin Verilog HDL tasarımlarını ise hazır olarak kullanmaktayız.



Şekil 2: VSCPU ile oluşturulan mikrodenetleyicinin blok diyagramı

Şekil 2'deki mikrodenetleyici tasarımı *Xilinx Artix-7 XC7A100T* FPGA tümdevresine yönelik olarak sentezlendiğinde rahatlıkla 100 MHz frekansa sahip bir saat ile çalıştırabilmektedir (ki bu saat frekansı bir mikrodenetleyici açısından oldukça yüksektir). Ayrıca kullandığımız tasarımların çoğu bu FPGA'deki yazmaçların yaklaşık %1'ini ve LUT'lerin yaklaşık %4 ila %10'unu, program ve veri belleği amaçlı kullandığımız dahili BRAM bloklarının da yaklaşık %3 ila %5'ini kullanmaktadır.

D. Toolchain

Bildirinin daha önceki kısımlarında da belirttiğimiz üzere VSCPU işlemcisine ait toolchain içerisinde tarafımızdan tasarlanıp gerçeklenen komut kümesi simülatörü, assembler ve C derleyici bulunmaktadır. Bu sayede, öğrenciler işlemciyi donanımsal olarak tasarlayıp saat bazlı benzetim ile doğrulamadan önce programlarını komut seti simülatörüyle de test edebilmektedirler. Bunun dışında assembler aracı, yazılan assembly programlarına karşılık düşen makine kodlarını oluşturmaktadır. C derleyicisi ise bütün C programlama dili standardının hepsi olmamakla birlikte işaretçi (pointer) aritmetiği, dizi işlemleri, fonksiyonlar, özyineleme (recursion), *for*/*while* döngüleri ve *int* veri tipini desteklemektedir. Desteklenen bu özellikler, öğrencilerin genel mikrodenetleyici uygulamaları yazmaları için fazlasıyla yeterlidir.

E. Bir Eğitim Aracı olarak VSCPU

VSCPU, iki farklı üniversitede 2012 yılından beri ilgili derslerde ve bu derslerin laboratuvar çalışmalarında etkin olarak kullanılmaktadır. Derslerde genel olarak VSCPU ve komut kümesi tanıtılmakta ve komut kümesindeki komutların içerdiği

mikroişlemler, kaç çevrim sürdüğü vb. konular detaylı olarak incelenmektedir. Ayrıca, öğrenciler VSCPU komut kümesinde karşılığı olmayan bazı aritmetik ve mantık işlemlerini VSCPU komut kümesindeki komutları kullanarak sentezlemeyi öğrenirler ve dolayısıyla işlemci komut kümesinin neden önemli olduğuna ilişkin fikir sahibi olurlar. Uygulama kısmında ise öğrencilerden VSCPU komut setini kullanarak belirli programları assembly seviyesinde yazmaları ve analiz etmeleri istenmektedir. Daha sonra, öğrencilerden yazmış oldukları bu programları önce VSCPU toolchain'indeki komut kümesi simülatörü sonra da Verilog HDL ile kendi tasarlayacakları VSCPU işlemcisini kullanarak bilgisayar ortamında doğrulamaları istenmektedir. Son olarak, VSCPU toolchain'indeki C derleyicisi ve assembler araçlarını kullanarak öğrencilerden C programları yazarak ve tasarladıkları VSCPU üzerinde de bu programları çalıştırarak çeşitli mikrodenetleyici uygulamalarını fiziksel olarak (board üzerinde) gerçeklemeleri istenmektedir.

III. Analiz: VSCPU ve PIC Mikrodenetleyicilerinin Karşılaştırılması

Bu kısımda, VSCPU komut kümesini Turing-complete (computationally universal) bir işlemcinin komut kümesiyle karşılaştırarak VSCPU'nun da bir Turing-complete işlemci olduğunu göstermek istedik. Bu amaçla, evrensel ve hem akademi hem de endüstride bilinirliği yüksek olan ve PIC-micro mimarisini temel alan PIC16F [3] serisi genel amaçlı mikrodenetleyicilerinde yer alan işlemciyi seçtik.

Tablo III'de VSCPU ve PIC16F87x arasında bazı temel ve teknik özellikleri içeren bir karşılaştırma verilmiştir. Tablodaki karşılaştırmaya ek olarak, şunları söyleyebiliriz:

- VSCPU PIC16F'den farklı olarak bit çözünürlüklü (bit-oriented) komut içermemektedir.
- VSCPU tek bir komut biçimine sahipken PIC16F'de farklı komut biçimleri bulunmaktadır.
- PIC16F'de PC yazmacı ve diğer özel fonksiyon yazmaçları VSCPU'dan farklı olarak işlemci içinde olmak yerine bellek haritasında özel adreslerde bulunmaktadır.

TABLO III: VSCPU-PIC16F karşılaştırması

	VSCPU	**PIC16F87x**
Bilgisayar Mimarisi	von Neumann/Harvard	Harvard
İndirgenmiş Komut Kümesi (RISC)	Evet	Evet
Komut Sayısı	16	35
Ortogonal (simetrik) Komutlar	Kısmen	Evet
Komut başına Saat Darbesi	2, 3 ya da 4	4
Saat Darbesi Frekansı	100 MHz	20 MHz
Komut Kelimesi Uzunluğu	32-bit	14-bit
Veri Kelimesi Uzunluğu	32-bit	8-bit
Komut Boru Hattı	Evet	Evet
G/Ç Erişimi	Bellek Haritalı	Bellek Haritalı
Kesme Desteği	Evet	Evet
Donanımsal Yığın Desteği	Hayır	Evet
Kayan Noktalı Aritmetik İşlem Desteği	Evet	Hayır

Bu analiz ile ilgili olarak yapmış olduğumuz diğer bir çalışma da PIC için yazılan programların VSCPU üzerinde de çalışabildiğini göstermek oldu. Şöyle ki; PIC mikrodenetleyicileri için yazılan C programlarını *Microchip MPLAB XC8* derleyicisiyle derledik ve derleyicinin ürettiği Intel HEX dosya formatındaki binary makine kodunu girdi olarak alıp

```
######################
running instruction at address 0x0000
######################
W:  0x0000
INDF: 0x0000
PCL:  0x0001
STATUS: 0x0018
FSR:  0x0000
PCLATH: 0x0000
######################
######################
running instruction at address 0x0001
######################
W:  0x0000
INDF: 0x0000
PCL:  0x0002
STATUS: 0x0018
FSR:  0x0000
PCLATH: 0x0000
######################
######################
running instruction at address 0x0002
######################
W:  0x0000
INDF: 0x0000
PCL:  0x00fc
STATUS: 0x0018
FSR:  0x0000
PCLATH: 0x0007
######################
######################
running instruction at address 0x07fc
######################
```

Liste 2: Cross-assembler betiğinin çalışırken ürettiği çıkıştan bir kesit

programın içerdiği PIC16F komutlarına karşılık düşen VSCPU komutlarını çıktı olarak veren bir Python betiği yazdık.

Şekil 3'de örnek bir C programı (Şekil 3a), bu programın PIC XC8 derleyicisiyle derlendiğinde elde edilen assembly programından bir kesit (Şekil 3b), PIC için üretilmiş Intel HEX dosya formatlı binary makine kodu (Şekil 3c) ve betiğin ürettiği VSCPU assembly programından bir kesit (Şekil 3d) gösterilmiştir.

Python dilinde yazmış olduğumuz cross-assembler betiği, ayrıca kendisine girdi olarak verilen PIC programının benzetimini yapabilmektedir. Buna ilişkin bir kesit Liste 2'de verilmiştir. Görüldüğü üzere, betik PIC16F makine kodunu girdi olarak alıp programın benzetimini yapmakta ve bunu yaparken her bir komut çalıştıktan sonra işlemciye ait önemli yazmaç ve bellek alanlarında meydana gelen değişiklikleri de göstermektedir. Betik çalışırken arka planda dinamik olarak her bir PIC16F komutuna karşılık gelen VSCPU komutlarını da sentezler.

Cross-assembler betiği kendisine girdi olarak verilen PIC16F programındaki her bir komutun VSCPU'daki eş değer komut bloğunu üretmek için bir dönüştürme şablonu kullanır. Buna göre her PIC16F komutuna karşılık gelen VSCPU komut bloğu betik içindeki ilgili şablon fonksiyonu tarafından üretilir. Tablo IV'de buna ilişkin üç farklı PIC16F komutuna karşılık gelen VSCPU komut bloklarına ait dönüştürme şablonu verilmiştir.

IV. Sonuç

Bu çalışmada, işlemci ve komut kümesinin eğitsel açıdan incelenmesi amacıyla ortaya koyduğumuz VSCPU işlemci mimarisi ve toolchain'i tanıtılmıştır. Bir işlemcinin komut

TABLO IV: PIC16F komut kümesine ait bazı komutların VSCPU komut kümesinde yer alan komutlar cinsinden karşılıkları

Komut Kelimesi	Mnemonic	Tanım	İşlev	Eş Değer VSCPU Komut Bloğu

00 0001 0xxx xxxx	CLRW	W yazmacını temizle	0 → W	// işlemi yap
				CPi W 0
				
				// Z bitini güncelle
				CPi <temp_1> 1
				SRLi <temp_1> 34
				NANDi <status> 251
				NAND <temp_1> <temp_1>
				NAND <status> <temp_1>

01 00bb bfff ffff	BCF f, b	f'ninci bellek adresindeki b'ninci biti sıfırla	0 → f<b>	CPi <temp_1> 1 SRLi <temp_1> (32+b) NAND <temp_1> <temp_1> NAND f <temp_1> NAND f f

10 0kkk kkkk kkkk	CALL k	k'nıncı bellek adresindeki altprogramı çağır	(PC)+1 → TOS,
			k → PC<10:0>,
			(PCLATH<4:3>) → PC<12:11>
				// işlemi yap
				CPi W 0
				
				// Z bitini güncelle
				CPi <temp_1> 1
				SRLi <temp_1> 34
				NANDi <status> 251
				NAND <temp_1> <temp_1>
				NAND <temp_1> <temp_1>
				NAND <status> <temp_1>

kümesinin detaylı olarak incelenmesi ve anlaşılması, öğrencilere daha sonra karşılaşacakları işlemcileri incelerken ve birbirleri arasında karşılaştırırken kolaylık sağlayacaktır. Biz bunu, öğrencilere VSCPU'yu PIC16F mikrodenetleyicisinde yer alan işlemciyle karşılaştırarak ve PIC16F programlarını VSCPU programlarına dönüştürerek uygulamalı olarak göstermiş olduk.

Gelecek çalışmalarımızda, VSCPU ve toolchan'ini geliştirerek eğitsel açıdan daha kapsamlı içerikler geliştirmek istiyoruz. Örneğin, öğrencilerin ilgili ders kapsamında VSCPU'nun C derleyicisini geliştirmeleri, VSCPU'nun komut kümesinin özelleştirilerek programa özel işlemci mimarilerinin geliştirilmesi vb.

BİLGİLENDİRME

Bu çalışma, TÜBİTAK tarafından 117E090 numaralı sözleşme kapsamında desteklenmektedir. Proje kapsamındaki bütün dokümantasyon, donanım ve yazılım kaynak kodları https://github.com/MC2SC web adresi altında sunulmuştur.

KAYNAKLAR

[1] A. Yıldız, H. F. Ugurdag, B. Aktemur, D. İskender, and S. Gören, "CPU design simplified," in *2018 3rd International Conference on Computer Science and Engineering (UBMK)*, Sep. 2018, pp. 630–632.

[2] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*. Elsevier, 2011.

[3] Microchip Technology Inc., "Microchip," https://www.microchip.com, 1997, [Online; accessed 24-February-2019].

```c
#include <xc.h>

void main(void) {

    int x = 0;
    x = 1;
    x = 5;
    x = x - 0x73;
    x = 0;
    x = 5;
    while(1);
    return;
}
```

(a)

```
Line   Address   Opcode   Label     DisAssy
1       0000      120A              BCF PCLATH, 0x4
2       0001      118A              BCF PCLATH, 0x3
3       0002      2FFC              GOTO 0x7FC
4       0003      3FFF              ADDLW 0xFF
...
2016    07DF      3FFF              ADDLW 0xFF
2017    07E0      01F0     main     CLRF __pcstackCOMMON
2018    07E1      01F1              CLRF 0x71
2019    07E2      3001              MOVLW 0x1
2020    07E3      00F0              MOVWF __pcstackCOMMON
2021    07E4      3000              MOVLW 0x0
2022    07E5      00F1              MOVWF 0x71
2023    07E6      3005              MOVLW 0x5
2024    07E7      00F0              MOVWF __pcstackCOMMON
...
```

(b)

```
:060000000A128A11FC2F18
:100FC000F001F1010130F0000030F1000530F000D7
:100FD0000030F10070088D3EF00071080318013EEA
:100FE000FF3EF100F001F1010530F0000030F100AA
:100FF000F82F0A128A11002883010A128A11E02FA1
:00000001FF
```

(c)

```
0: BZJi 3 17 // goto main
1: 0 // sp
2: 0 // bp
3: 0 // zero
4: 4294967295 // negative one
5: 0 // cpu specific data 1
6: 0 // cpu specific data 2
7: 0 // cpu specific data 3
8: 0 // cpu specific data 4
9: 0 // cpu specific data 5
10: 0 // cpu specific data 6
11: 0 // general-purpose data 1
12: 0 // general-purpose data 2
13: 0 // general-purpose data 3
14: 0 // general-purpose data 4
15: 0 // general-purpose data 5
16: 0 // general-purpose data 6
// 0x120a @ 0x0000
17: CPi 8193 1
18: SRLi 8193 36
19: NAND 8193 8193
20: NAND 8214 8193
21: NAND 8214 8214
...
// memory addresses related to PIC-to-VSCPU conversion
8192: 0 // wreg
8193: 0 // temp_1
8194: 0 // temp_2
8195: 8196 // hardware stack pointer
8206: 0 // PCL
8207: 24 // STATUS register Power-on Reset value
8214: 0 // PCLATH
8215: 0 // INTCON
// 8716: START OF PCLATH ADDRESS TRANSLATION TABLE
// 8780: START OF PCL ADDRESS TRANSLATION TABLE
// END OF THE PROGRAM
```

(d)

Şekil 3: *(a)* PIC işlemcisi için yazılmış örnek bir C programı. *(b)* PIC XC8 derleyicisiyle elde edilen assembly programı. *(c)* Assembly programına karşılık gelen Intel HEX dosya formatındaki makine kodu. *(d)* Betik tarafından üretilen VSCPU assembly programı.