Assembly Dili'nin Yüksek Seviyeli Bir Dil Olan C İle Bağlantısı

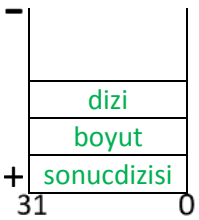
Hazırlayanlar: Arş. Grv. Furkan ÇAKMAK (furkan@ce.yildiz.edu.tr, Yrd. Doç. Dr. Ahmet Tevfik İNAN (tevfik@ce.yildiz.edu.tr)
Yıldız Teknik Üniversitesi, Bilgisayar Mühendisliği Bölümü
(http://www.ce.yildiz.edu.tr)
İstanbul

Bu kılavuzda Assembly programlama dilinin yüksek seviyeli bir dil olan C ile nasıl birlikte kullanılacağı detaylı ve uygulamalı olarak anlatılmıştır. Uygulamalar farklı platformlar kullanılarak ve farklı örnek problemler ile gerçekleştirilmiştir. Zaman zaman anlatımı kolaylaştırması için ekran görüntülerine, yığın organizasyonlarına ve program parçacıklarına yer verilmiştir.

Uygulama Öncesi Temel Bilgiler

- Assembly yordamlarının C fonksiyonu ile benzer kullanılabileceği gibi aynı dosya içerisinde hem assembly hem de C kodu yazılabilmektedir. Ancak platform farklılıklarından kaynaklanan yazım değişiklikleri bilinmelidir.
- C ve Assembly dillerinde yazılan kodların aynı tipteki büyüklükte (32-bit, 64-bit, vb.) derlenmesi veya en azından üretecekleri nesne dosyalarının aynı tipte olması sağlanmalıdır.
- Eğer bu kılavuzda ilk iki uygulamasında olduğu gibi fonksiyon kullanımı tercih edilecekse parametre aktarmanın ne şekilde gerçekleştirileceği önem arz etmektedir.
    - C fonksiyonundan Assembly yordamı çağırıldığında aktarılan parametreler; aktarım sırasının tam tersi olacak şekilde (sağdan sola doğru) yığına atılırlar. Yığından aldığı verileri işleyen Assembly yordamı eğer tek bir değer döndürecekse akümülatör yazmacı (kullanılan platforma göre değişiklik göstermekle birlikte AX (16-bit), EAX (32-bit) veya RAX (64-bit) üzerinden değer döndürmektedir. Örnek 1'de integer'ın 4 byte olduğu bir platform için yazılmış bir C fonksiyonu prototipi ve bu prototipin çağırılması sonucu parametrelerin ne şekilde yığına atılacağı gösterilmiştir. Fonksiyon bir tamsayı değeri döndüreceği için bu da EAX yazmacı üzerinden gerçekleştirilecektir.

      Örnek 1: int Toplama(int* dizi, int boyut, int* sonucdizisi);



Şekil 1: Örnek 1 Fonksiyonunun Çağırılması Sonucu Parametrelerin Yığına Konulma Sırası

- Çağırılan yordamların "near" tipinde olduğu bilinmeli ve yığına bu sebeple bir göreceli konum (offset) değerinin atıldığı unutulmamalıdır.
- C programlama dilinin yapısı gereği yığın üzerinden parametre olarak gönderilen değerler, yine C tarafında yığından kaldırılmaktadır. Bu sebeple Assembly yordamından dönerken bu parametrelerin kaldırılması ile ilgili herhangi bir işlem yapılmamalıdır.
- Farklı platformlarda değişiklikler göstermekle birlikte unutulmamalıdır ki; bir assembly program (COM tipinde hepsi bir arada olsa bile) veri, yığın ve kod kesimlerinden oluşur. Eğer bir assembly yordamı, bir C fonksiyonu tarafından çağırıldıysa, yığın olarak çağıran programın

(yani C programının) yığını kullanılır. Eğer yordamda veri kullanımına ihtiyaç duyulmuşsa, ihtiyaç duyulan veriler, veri kesiminde tanımlandıktan sonra kod kesiminde kullanılabilir.

- Programlar içerisinde kullanılan değişkenler, bellekte birer adresi göstermektedir. Bu sebeple inline assembly kodu yazarken C değişkenleri olduğu gibi kullanılabilir (okunup, yazılabilinir). Yani ortak veri kesimi kullanımı gerçekleştirilmektedir.
- Assembly yordamına int'den büyük değerler (double vb.) gönderilmek istenirse, unutulmamalıdır ki, veri int'den büyük tanımlı olsa bile o verinin bellekteki adresi bir int boyutundadır. Bu da, istenilen her büyüklükteki verinin (kaynakların kapasitesi göz önünde bulundurularak) assembly yordamına gönderilebilmesine imkan sağlamaktadır.

1 - Linux (Ubuntu) Ortamında Assembly ve C Dillerinin Birlikte Kullanımı

Bu uygulamada farklı bir assembler olan **nasm** (The Netwide Assembler http://www.nasm.us/) kullanılarak assembly kodları derlenmiş, ardından C nesne kodları ile birlikte linklenerek çalıştırılabilir dosya üretilmiştir.

Öncelikle kodlama yapılacak dosyaların bulunacağı bir klasör Masaüstü'nde oluşturulmuştur.

```
$ mkdir ~/Dekstop/casm
```

Ardından ornek1.c ve ornek1.asm dosyaları oluşturulmuştur.

```
$ cd ~/Dekstop/casm
$ > ornek1.c
$ > ornek1.asm
```

Bu örnekte C programında tanımlanmış **n** elemanlı bir dizi içerinde bulunan değerlerin çağırılan bir Assembly yordamı tarafından toplanarak sonucunun döndürülmesi sağlanmıştır. Öncelikle C kodlarını yazmak için gedit uygulamasıyla C dosyasını açalım.

```
$ gedit ornek1.c
```

Yazılan C dosyası aşağıda verildiği gibidir.

ornek1.c

```
#include <stdio.h>
#include <stdlib.h>
extern int Topla(int *, int);

int main(){
	int i, top, n;
	printf("Kaca kadar sayilari toplamak istiyorsunuz: ");
	scanf("%d", &n);
	int dizi[n];
	for(i = 0; i < n; i++)
		dizi[i] = i + 1;
	top = Topla(dizi, n); //Assembly yordam çağırma satırı
	printf("Toplam: %d\n", top);
	system("PAUSE");
	return 0;
}
```

"ornek1.c" dosyası içerisinde "Topla" isimli bir assembly yordamı kullanılmıştır. Yordamın kodu bu dosya içerisinde olmadığından derleyiciyi uyarmak için yordamın prototipi yazılırken "extern" ön eki kullanılmıştır. Toplama işlemini gerçekleştirecek Assembly dosyası ornek1.asm ise;

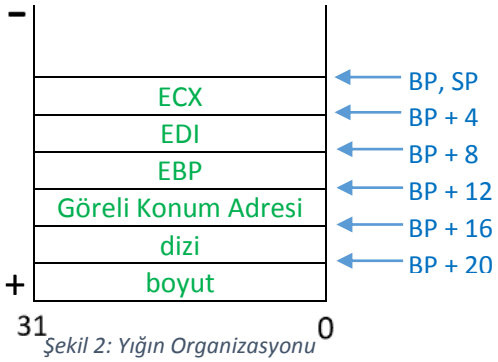
```
$ gedit ornek1.asm
```

Koduyla oluşturulmuş ve içeriği aşağıdaki gibi yazılmıştır.

ornek1.asm

```
section .data

section .text

global Topla

Topla:
       PUSH EBP
       PUSH EDI
       PUSH ECX
       MOV EBP, ESP
       MOV  EDI, [EBP+16]   ; Dizinin adresine erisilir.
       MOV  ECX, [EBP+20]   ; Dizinin boyutu olan n degerine erisilir.
       XOR EAX, EAX
L1:    ADD EAX, [EDI]
       ADD EDI, 4           ; Dizi integer (4 byte) tanimli oldugu icin bir sonraki
       LOOP L1              ; elemana erisim icin 4 eklenir.
       POP ECX
       POP EDI
       POP EBP
       RET
```

Assembly kodu içerisinde veri tanımı yapılmadığı için data kısımı/kesimi boş bırakılmıştır. "Topla" yordamına ':' kullanılarak başlanmış RET ifadesi ile de bitirilmiştir. Yordama girildiğinde öncelikle kullanılacak yazmaçlar yığına atılmış ve göreceli olarak yığın üzerinden gelen parametrelere erişildikten sonra işlem gerçekleştirilmiştir. İşlemin sonucunun akümülatör (EAX) yazmacı üzerinde oluşması sağlanmıştır. Çünkü daha önceden de bahsedildiği gibi RET komutuyla birlikte yordamdan C programına dönüldüğünde bulunan sonuç akümülatör yazmacı üzerinden beklenmektedir. Kodun daha iyi anlaşılabilmesi için yığın organizasyonu Şekil 2'de verilmiştir.



Şekil 2: Yığın Organizasyonu

Kodları derlemek için bir "makefile" oluşturulmuştur.

```
$ gedit makefile
```

"makefile" dosyası içerisinde öncelikle Assembly source kodundan elf (Executable and Linkable Format) formatında obje kodu üretilmesi için;

```
nasm -f elf32 -o ornek1.o ornek1.asm
```

satırı yazılmıştır. Ardından C kodunu derleyecek ve Assembly kodu ile linkledikten sonra çalıştırılabilir dosya üretecek;

```
gcc –m32 ornek1.c ornek1.o -o combine_casm
```

kodu yazılmıştır.

64-bit'lik linux platform üzerinde gcc derleyicisinin nesne kodu üretmesi için m32 parametresi kullanılmıştır. Eğer sisteminizde yüklü değilse öncelikle;

```
$ sudo apt-get install gcc-multilib
```

komutu ile gcc derleyicisinin çoklu derleme opsiyonu yüklenmelidir. Bu sayede m32 parametresi kullanılabilecektir.
"nasm" ile Assembly kodunu derlerken kullanılan elf32 parametresi ise üreteceği nesne dosyasının 32-bit'lik olmasını sağlamak içindir.
Programların derlenmesi için ilgili dizine gidildikten sonra "make" komutu çalıştırılmalı, ardından oluşan combine_casm çalıştırılabilir dosyası aşağıda verildiği gibi çalıştırılmalıdır.

```
$ cd ~/Desktop/casm
$ make
$ ./combine_casm
```

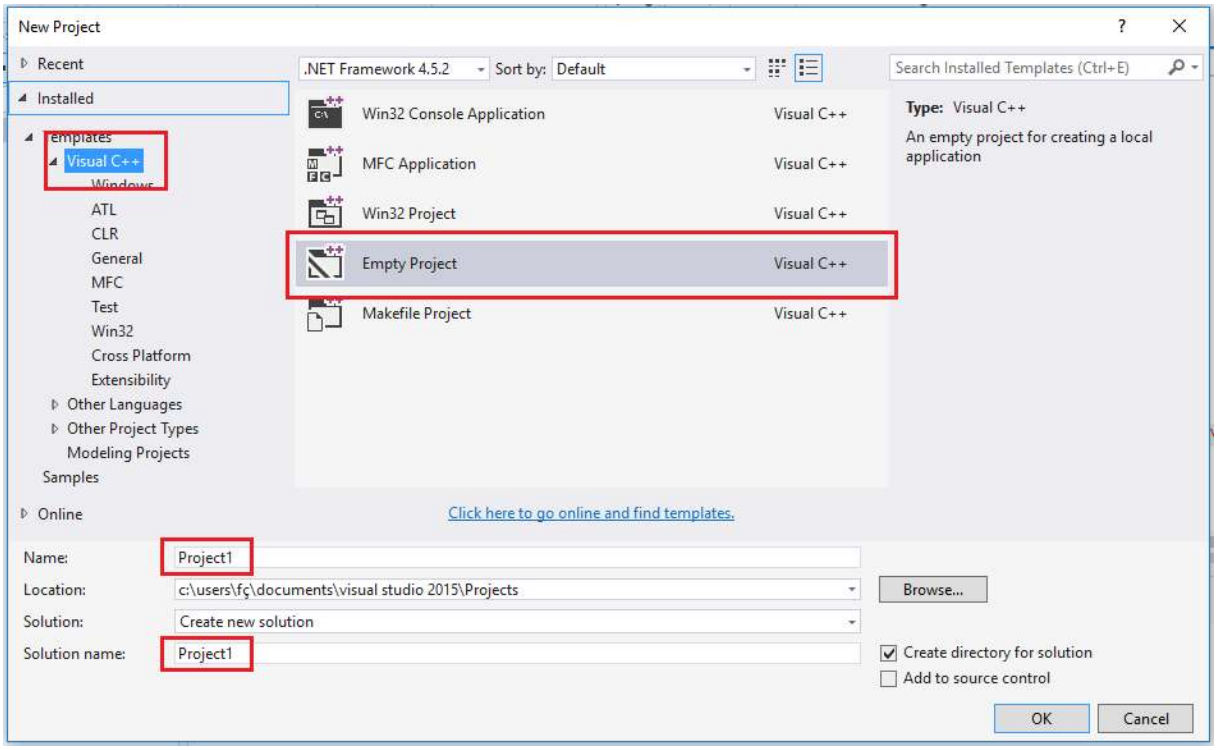
Bu işlem gerçekleştirildiğinde program terminal ekranında çalıştırılacak ve sizden istediği "n" değeri ile işlemi gerçekleştirecektir.

Windows Ortamında Visual Studio 2015 Kullanılarak C Programı İçerisinden Assembly Yordamı Çağırma

Bu başlıkta, bir önceki başlıkta yapılan işlemler, Windows ortamında kullanıcı dostu bir IDE üzerinde gerçekleştirilecektir.

Örnek olarak Worst-Case durumunda Selection Sort yapan bir C ve bir Assembly fonksiyonu yazılacaktır. C kodu içerisinden her ikisi de çağırılacak ve çalışmaları debug kullanılarak incelenecektir.

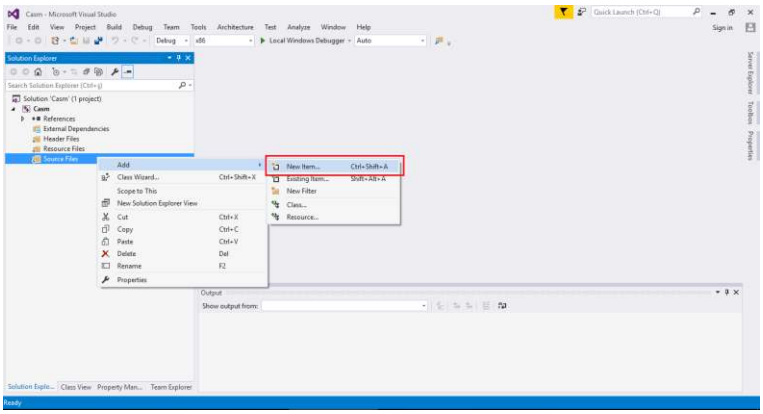
Öncelikle Visual Studio 2015 ortamı çalıştırılır. File -> New -> Project modülü tıklanır ve yeni proje oluşturmak için gerekli ekran ile karşılaşılır (Şekil 3).



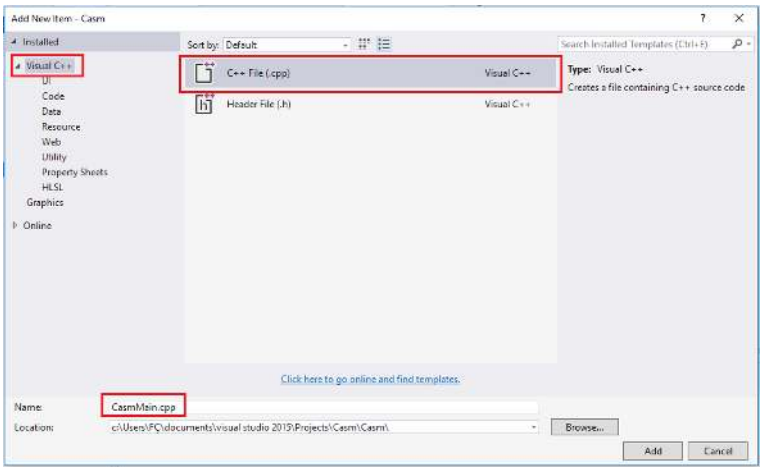
Şekil 3: Yeni Proje Oluşturma Ekranı

Şekil 3'te görüldüğü gibi açılan segmede "Visual C++", ardından "Empty Project" seçildikten sonra proje ve çözüm ismi girilerek boş bir C++ projesi oluşturulur. Unutulmamalıdır ki C++ derleyicileri aynı zamanda C kodlarını da derlemektedir. Bu bakımdan kodumunuz C++ olarak kaydedilmesinde ve C++ derleyicisi tarafından derlenmesinde bir sıkıntı olmayacaktır.

Oluşturulan boş projenin "Source Files" kısmı sağ tıklanır ve Şekil 4'te görülebileceği gibi bir C++ kaynak dosyası oluşturulur.



(a)



(b)

Şekil 4: (a) Yeni Madde Seçimi İşlemi, (b) C++ File (.cpp) Dosyası Adıyla Birlikte Oluşturma İşlemi

Oluşturulan ve main fonksiyonu ihtiva eden C programı aşağıdaki gibidir.

CasmMain.cpp

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>

extern "C" void SelSortASM(int *, int);
void SelSortC(int *, int);
const int n = 100000;

int main(){
	int dizi[n];
	int i;
	double sure;
	clock_t bas, bit;
	for (i = 0; i < n; i++)
		dizi[i] = i + 1;
	bas = clock();
	SelSortC(dizi, n);
	//SelSortASM(dizi, c);
	bit = clock();
	sure = (double)(bit - bas) / CLOCKS_PER_SEC;
	printf("Toplam sure: %.2f sn\n", sure);
	printf("Dizinin ilk elemani: %d, son elemani: %d\n", dizi[0], dizi[n - 1]);
	system("PAUSE");
	return 0;
}
```

Kod içerisinde 1'den "n" değerine kadar ardışık elemanları olan bir dizi tanımlanmıştır. Küçükten büyüğe sıralı olan bu dizinin hem "SelSortC()" hem de "SelSortASM()" fonksiyonları ile sıralanması sağlanmıştır. Yazılan fonksiyonların süre olarak karşılaştırılabilmesi için "time.h" kütüphanesinin "clock_t" veri tipi ve "clock()" fonksiyonları kullanılmıştır.

Öncelikle aşağıda verilen C fonksiyonu ile Selection Sort işleminin gerçekleştirilmesi sağlanmıştır.

```
void SelSortC(int *sdizi, int n) {
	int i, j, tmp, max;
	for (i = 0; i < n - 1; i++) {
		max = i;
		for (j = i + 1; j < n; j++)
			if (sdizi[j] > sdizi[max])
				max = j;
```

```
                tmp = sdizi[max];
                sdizi[max] = sdizi[i];
                sdizi[i] = tmp;
        }
}
```

Prototipten de anlaşılabileceği üzere fonksiyon, sıralanmak istenilen dizinin adresini ve boyutunu alarak, ek bir dizi kullanmadan sıralama sonucunu aynı bellek alanında üretmek üzere yazılmıştır. Bu fonksiyon yukarıda oluşturulmuş "CasmMain.cpp" dosyasının "main()" fonksiyondan sonraki kısmına eklenmiştir.

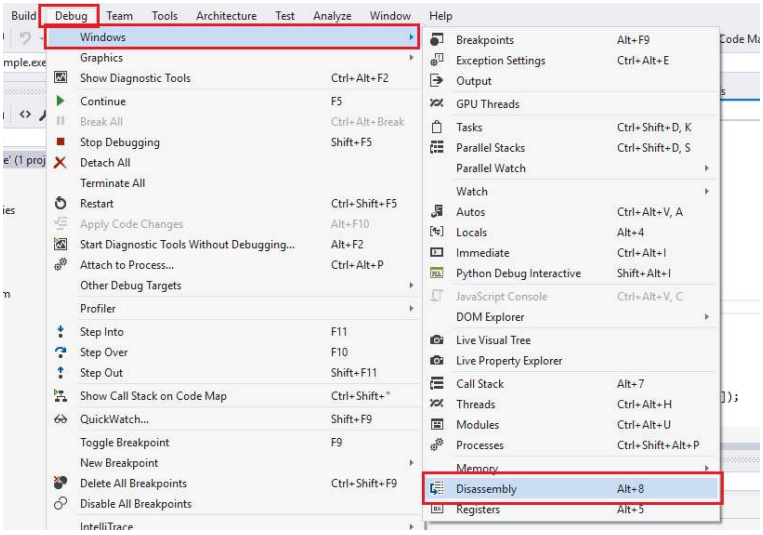
C kodunun Disassembly edilebilmesi için, "main()" fonksiyon içerisinde sıralama fonksiyonunun çağırıldığı satıra bir "break point" konulmuş ve "F5" tuşuna basılarak debug işlemi başlatılmıştır. Debug işlemi "break point" konulan noktada durduktan sonra Debug -> Windows -> Disassembly yolu izlenerek Şekil 5'te gösterildiği gibi C fonksiyonuna karşılık gelen Assembly kodu gözlemlenebilmiştir.

Debug fonsiyonları (F11: Adım adım çalıştır (trace), F10: fonksiyon ve döngüleri atlayarak çalıştır (proceed), F5: Tamamını çalıştır) kullanılarak kod adım adım çalıştırılıp C kodunun derleyici tarafından ne şekilde assembly'e dönüştürüldüğü görülebilir.

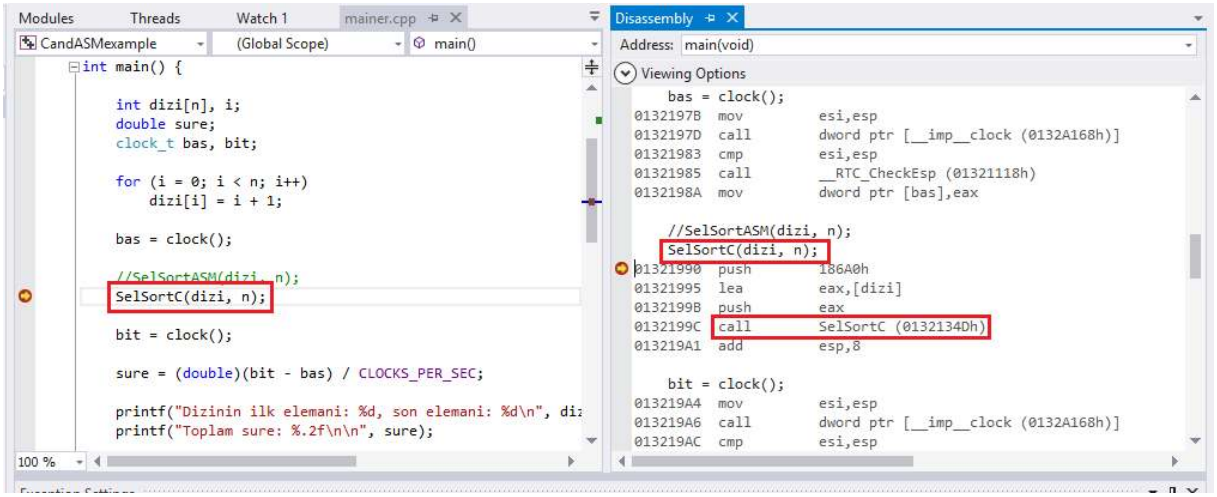
Ancak bu işlemlerin "Debug" modda yapıldığını hatırlatmak ta ve "Release" modda yapıldığında disassembly aşamasında daha az assembly kodu ile karşılaşacağını belirtmekte fayda var.

Şekil 5-b'ye bakıldığında öncelikle kullanılacak parametreler yığına atılmakta ve "call" komutu ile ilgili yordam çağırılmaktadır. Ancak bir diğer dikkat edilmesi gereken husus da fonsiyon çağırımı tamamlandığında IP yazmacının göstereceği koddur ki bu örneğimizde "add esp,8" şeklindedir. Daha önceden de belirtildiği gibi, C de fonksiyon kullanımında, gönderilen parametrelerin, gönderen tarafından kaldırılması gerekmektedir. Burada da SP yazmaç değeri 8 arttırılarak bu yapılmaya çalışılmıştır. Burada kesinlikle programcının ekstra bir müdahalesi yoktur, olmamalıdır. C derleyicileri bu işlemleri otomatik olarak gerçekleştirecek şekilde tasarlanmışlardır.

İstenildiği takdirde debug fonksiyonları kullanılarak kod adım adım incelenebilir. Ancak bu işlem uzun süreceği için burada gerçekleştirilmeyecek, yazılan Assembly yordamının incelenmesine geçilecektir.



(a)



(b)

Şekil 5: (a) Disassembly Görüntüleme Başlatma Butonu, (b) C Kodunun Derleyici Tarafından Üretilen Assembly Karşılığı

Assembly yordamının yazılacağı dosyayı oluşturmak için yapılması gerekenler biraz daha farklıdır. Şöyle ki, Visual Studio ortamında projeler build edildiğinde her bir dosya tipinin ne şekilde derleneceği bilinmektedir ve kullanıcının ayrıca bunu bildirmesine gerek yoktur. Ancak Assembly dosyalarının ne şekilde derleneceğini proje ayarlarında bildirmemiz gerekmektedir.

Yine “Source Files” üzerine sağ tıklayarak ismi “casm.asm” olacak şekilde boş bir dosya oluşturulması sağlanmalıdır. Bu dosyanın içerisine “SelSortAsm()” yordamını barındıracak aşağıda verilen Assembly kodu yerleştirilecektir.

casm.asm

```
.586
.model flat, c
.stack 100h
.data

.code

SelSortASM    PROC NEAR

              PUSH EBP
              PUSH ECX
              PUSH EDI
              PUSH ESI
              PUSH EAX
              PUSH EBX
              PUSH EDX

              MOV EBP, ESP
              MOV EDI, [EBP+32]
              MOV ECX, [EBP+36]

              DEC ECX
              MOV EAX, EDI
dis_don:      MOV EBX, ECX
              MOV ESI, EAX
              MOV EDI, EAX
              MOV EDX, [EDI]
ic_don:       ADD ESI, 4
              CMP [ESI], EDX
```

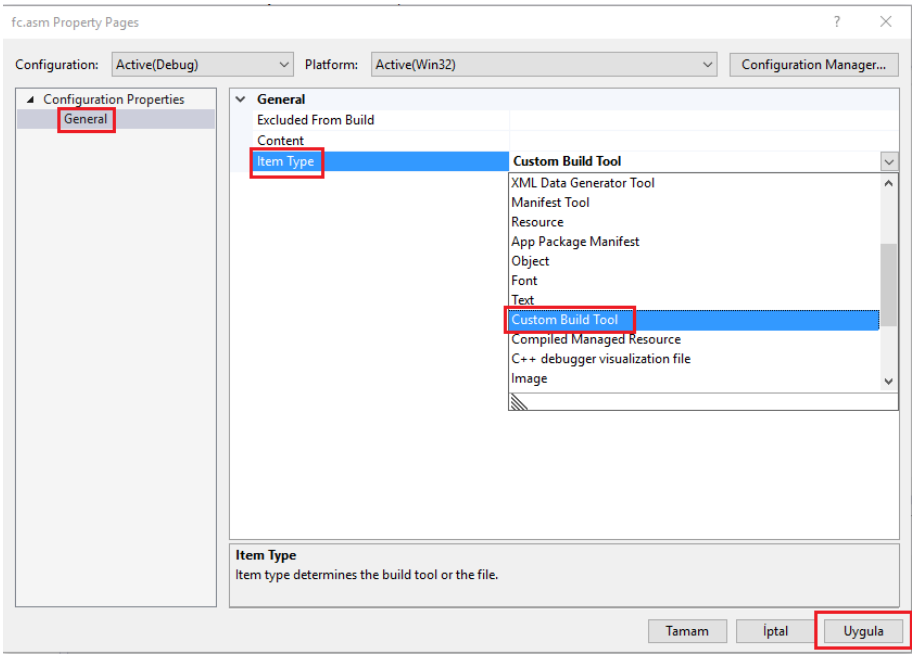
```
            JGE ATLA
            MOV EDI, ESI
            MOV EDX, [EDI]
ATLA:       DEC EBX
            JNZ ic_don
            MOV EDX, [ESI]
            XCHG EDX, [EDI]
            MOV [ESI], EDX
            LOOP dis_don

            POP EDX
            POP EBX
            POP EAX
            POP ESI
            POP EDI
            POP ECX
            POP EBP
            RET
SelSortASM  ENDP
            END
```

Hiç kuşkusuz Selection Sort algoritması farklı iyileştirmeler yapılarak daha optimize bir şekilde çalıştırılabilir. Ancak buradaki amaç varolan bir kodu optimize etmek değil Assembly yordamlarının C programlarından çağırılmasının sağlanmasıdır.

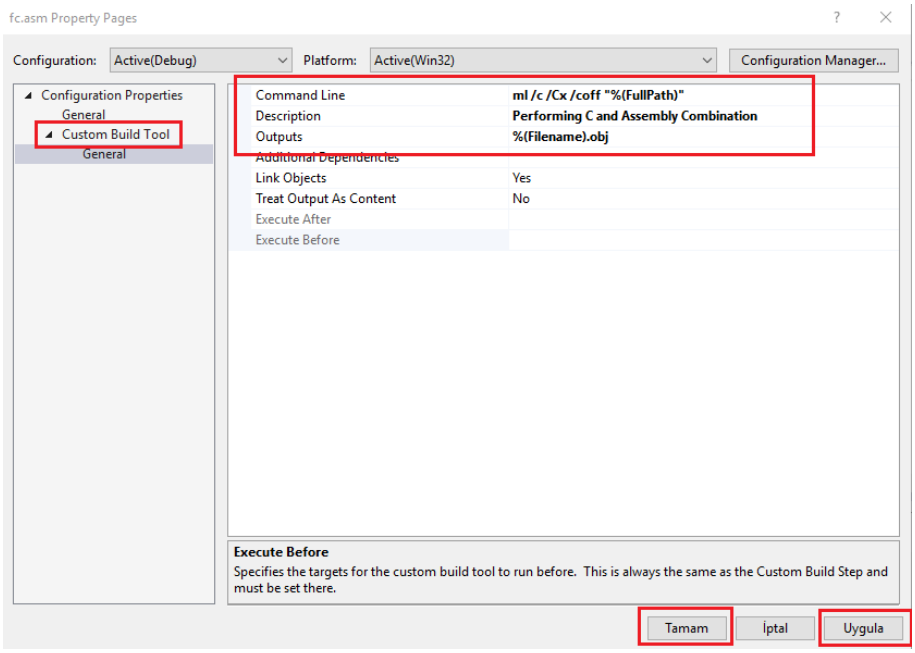
"Main()" fonksiyon içerisinden C'de yazılmış sıralama fonksiyonunu çağıran satır kapatılıp Assembly'de yazılmış fonksiyon satırı açılmalıdır.

Oluşturulan "casm.asm" dosyasının ne şekilde derleneceğini söylemek için öncelikle proje bölmesindeki dosyaya sağ tıklanır ve "Properties/Özellikler" seçilir. Şekil 6'da gösterildiği gibi açılan ekranda "Item Type" bölümünde "Custom Build Tool" segmesi seçilir ve "Apply/Uygula" butonuna basılır. Bu butona basıldıktan sonra aynı ekranda "Custom Build Tool" segmesi açılacaktır. Bu segmenin altında bulunan "General" segmesine tıklanarak custom build'in ne şekilde gerçekleştirileceğinin söyleneceği ekrana geçilmelidir. Bu ekranın bir görüntüsüne Şekil 7'de yer verilmiştir.

Açılan ekranda "Command Line" kısmına **ml /c /Cx /coff "%(FullPath)"** komutu, "Outputs" kısmına da **%(Filename).obj** ismi yazılmalıdır. Bu komutları yazarak oluşturduğumuz Assembly dosyasının ml (Miscrosoft Assembler) ile derlenmesini ve sonucunda aynı ismi taşıyan bir obje dosyası oluşturulmasını sağlamış oluyoruz.



Şekil 6: Dosya Özellikleri (Properties) Sayfası



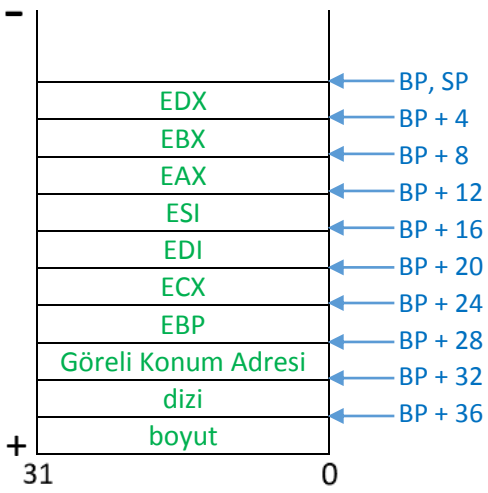
Şekil 7: Custom Build Ayarlarının Yapılacağı Sayfa

Komut debug edilmeye başlanmadan önce Şekil 8'de verilen yığın organizasyonuna bir göz atılabilir.

Ardından yukarıda anlatıldığı gibi adım adım debug işlemi gerçekleştirilerek yazılan Assembly kodun ne şekilde çalıştığı, disassembly edilmiş C koduyla ne farklılıklar gösterdiği incelenebilir.

Farklı örnek problemler hem C fonksiyonu hem de Assembly yordamı olarak yazılarak hız ve performans karşılaştırması bu şekilde gerçekleştirilebilir.

Bu anlatılanlar öğrenildikten sonra yazılacak aynı işi yapan Assembly yordamlarının en az C fonksiyonları kadar hızlı çalışması beklenmektedir. Çünkü yazılan C fonksiyonları disassembly edildiğinde zaten assembly karşılıkları görülebilecektir.



Şekil 8: Yığın Organizasyonu

Visual Studio 2015 Ortamında C Dosyası İçerisinde Inline Assembly Komutu Yazma Örneği

Bu başlık altında belki en sık kullanılan C/Assembly kod birleştirme yöntemi olan inline assembly kodu yazma incelenmiştir. Yani aynı dosya içerisinde bir C fonksiyonunda Assembly kodu yazarak işlemlere devam edilmeyi sağlayan yapı incelenmiştir.

Bunun için bir üst başlıkta anlatıldığı şekilde yeni bir proje ve bu proje içerisinde bir tane C++ kaynak kod dosyası oluşturulmuştur.

Visual Studio ortamında c kodunun içerisinde;

```
__asm {
       ;Assembly kodları
}
```

komutu kullanılarak Assembly kodu yazılması sağlanmaktadır. Ancak farklı tür IDE ve platformlarda farklı notasyonların (Intel ve AT&T olarak yaygın kullanılan 2 notasyon avrdır. Ders kapsamında Intel notasyonu kullanılmıştır.) kullanılabileceğini unutmamak gerekir.

C'de bir dizide bulunan elemanları dizinin ortasına göre tersine çevirmek kolay bir işlem sayılır. Ancak integer (32-bit) tanımlı bir dizide, dizinin her bir elemanının ikili (binary) gösterimindeki digit'lerin tersine çevrilmesi o kadar basit bir problem değildir. Bu tarz Assembly dilinde yapılması C diline göre daha kolay olan işlemlerin inline Assembly kodu olarak yazılması hiç kuşkusuz işleyişi oldukça kolaylaştıracaktır. Bu başlık altında yukarıda anlatılan örnek gerçekleştirilmiştir. Sorunun daha iyi anlaşılabilmesi için ikili digit'ler üzerinde tersine çevirme (reverse) işleminin nasıl olduğu aşağıda örnek ile verilmiştir.

Örnek: $1101011110000\,1 \xrightarrow{reverse} 1000011110101\,1$

Oluşturulan proje içerisinde "inline.cpp" olarak isimlendirilen C++ kaynak dosyası içerisine aşağıda verilen kod yazılmıştır.

Kod içerisinde dizinin elemanlarının bitleri tersine çevrilmeden önce hexadecimal olarak yazdırılması gerçekleşmiş, ardından assembly kodu yazılmış ve elemanlar tekrar ekrana yazdırılmıştır. Böylece yazılan Assembly kodunun doğru çalışıp çalışmadığı ekran çıktılarından anlaşılabilecektir.

Görülebileceği gibi C değişkenleri, Assembly içerisinde kullanılmıştır. Uygulama öncesi temel bilgiler bölümünde de bahsedildiği gibi değişkenler bellekte adres gösteren birimlerdir. Derleyici kodu derlerken onun için bütün değişkenler adreslerden ibarettir. Böyle olunca C değişkenlerinin Assembly içerisinde kullanımı sorun teşkil etmeyecektir.

inline.cpp

```
#include <stdio.h>
#include <stdlib.h>
const int n = 10;

int main() {
	int i, dizi[n];
	for (i = 0; i < n; i++) {
		dizi[i] = i + 1;
		printf("%8x, ", dizi[i]);
	}
	printf("\n\n");

	__asm {
		MOV ECX, n
		XOR ESI, ESI

	L2:	MOV EAX, dizi[ESI]
		PUSH ECX
		MOV ECX, 32

	L1:	SHR EAX, 1
		RCL EBX, 1
		LOOP L1
		POP ECX

		MOV dizi[ESI], EBX
		ADD ESI, 4
		LOOP L2
	}

	for (i = 0; i < n; i++)
		printf("%8x, ", dizi[i]);
	system("PAUSE");
	return 0;
}
```

Neticeler

- C programları içerisinde hıza ihtiyaç duyulduğu zaman Assembly dili rahat bir şekilde kullanılabilir.
- Assembly kodlarının yordamlar olarak hazırlanması ve C dosyalarından fonksiyon çağırır gibi çağırılması her ne kadar modüler programlama açısından elverişli ve tercih sebebi olsa da bazı durumlarda inline assembly kodu yazmanın da sağladığı avantajlar göz ardı edilemez.
- Günümüzde büyük veri analizlerinin gerçekleştirilmesi için günlerce çalışma süresi olan kodlar çalıştırılmaktadır. Bazı durumlarda %10 bile performans artışının çok faydalı olabileceği göz önünde bulundurulmalı ve Assembly kartını masaya koymak için her daim güncel ve hazır olunması iyi olacaktır.
- Assembly dilinin de yüksek seviyeli dillerde olduğu gibi bir çok kütüphanesi olduğu ve bu kütüphanelerde bir çok ihtiyaç duyulan program, yordam seviyesinde kullanıcıya sunulduğu unutulmamalıdır. C içerisinde kullanılan printf ve scanf gibi fonksiyonların da bir kütüphane tarafından sağlandığı, bu kütüphaneyi kullanmadan ekrana bir yazı yazmanın veya okumanın

ne kadar zor olacağı düşünüldüğünde Assembly için hazırlanmış çeşitli kütüphanelerin ihtiyaç duyulduğunda araştırılarak kullanılması belki hayat kurtarabilir ☺