Mikroişlemciler, IV. Uygulama Yönergesi

Amaç: Tümleşik Geliştirme Ortamı kullanarak C++ dilinin gelişmiş komut satırı arguman işleme ve giriş/çıkış özelliklerini Assembly dili programların gerek duyduğu veriyi okuma/yazma için kullanma.

Hedef: Visual Studio'da (VS) geliştirilen bir C++ program ile komut satırından 1. arguman olarak adı verilen dosyayı açıp okuyarak bir veri yapısına sakladıktan sonra bu veri yapısı ve gerekli diğer argumanları ile Assembly dilinde yazılmış olan bir fonksiyonu fastcall arguman aktarma ile çağırmak. Fonksiyon geri dönüş değerini ekranda gösterdikten sonra geri aktarılan fonksiyon tarafından işlendiği varsayılan veri yapısını 2. arguman olarak adı verilen dosyaya saklamak.

Bu uygulamada, 3. uygulama yönergesinde açıklanan VS'nun güncel bir sürümünü bilgisayarınıza indirerek uygun seçenekleri (en az C++) ile kurunuz.

VS'da komut satırından arguman alabilen ve Assembly dilinde fonksiyon çağırabilen bir C++ konsol uygulaması oluşturmak için, **Dosya** -> **Yeni** -> **Proje** (Ctrl+Shift+N) ile çıkan proje seçenekleri içindeki **Yüklü** -> **Şablonlar** -> **VC++** altından **Win32 Konsol Uygulaması**'nı seçmeniz gerekmektedir. Projeyi oluşturuken kullanılan varsayılan ad (ConsoleApplication1) yerine, daha sonra çözümü açtığınızda projenizi kolay hatırlamanızı sağlayacak şekilde yaptığınız işe uygun bir ad vermeniz gerekir. Bu uygulamada C++ ve Assembly kullanıldığı için, proje adı **cpp-asm** olarak seçilmiştir. Seçtiğiniz dosya adı ile oluşturulacak proje dosyası detaylarının sorulduğu arayüzde, **Son** ile varsayılan proje dosya özelliklerini kabul etmek yerine, **ileri** seçeneği ile ilerlediğinizde karşınıza çıkan seçeneklerden seçili çıkan **Önceden derlenmiş üst bilgi** (Precompiled headers) seçeneğinin kapatılması ve **Boş proje**'nin seçilmiş olması, derleme işleminin daha hızlı yapılabilmesi ve daha küçük çalıştırılabilir kodun üretilebilmesi için gereklidir.

Konsol uygulaması projenize, komut satırı okuma ve dosya açma, okuma/yazma ve kapatma işlemlerini yapacak bir C++ kodu eklemek için, C++ proje dosyası üzerinde sağ tıklayarak **Ekle** -> **Yeni Öğe** seçeneği ile karşınıza çıkan listedeki **C++ Dosyası (.cpp)** seçeneği için varsayılan ad (Kaynak) uygun bir dosya adı ile değiştirilerek yeni C++ dosyası projeye eklenmelidir. Bu yönergede **cpp-kod.cpp** adı ile C++ dosya oluşturulmuştur.Yeni oluşturduğunuz C++ dosyası için gerekli giriş/çıkış işlemlerini yapan örnek bir C++ dosya'yı indirerek kullanabilirsiniz.

Benzer şekilde konsol uygulaması projenize, kendisine aktarılacak veriyi işleyen bir Assembly kodu eklemek için, C++ proje dosyası üzerinde sağ tıklayarak **Ekle** -> **Yeni Öğe** seçeneği ile karşınıza çıkan listedeki **Visual C++** -> **Yardımcı Program** altında çıkan listedeki **Metin Dosyası (.txt)** seçeneği için varsayılan adı (Text.txt) uygun bir dosya adı ile değiştirilerek yeni Assembly dosya projeye eklenmelidir. Bu yönergede **asm-kod.asm** adı ile Assembly dosya oluşturulmuştur. Kendisine göstericisi ile aktarılan bir tamsayı dizisinin toplamını bularak geri döndüren örnek bir Assembly dosya'yı indirerek kullanabilirsiniz.

Uyarı : VS yüksek seviyeli diller için bir geliştirme ortamı olduğundan ve varsayılan derleme seçenekleri içinde Assembly dili kaynak dosyalarına ilişkin derleme kuralları varsayılan olarak açık olmadığı için, Assembly dilindeki kaynak dosyalarını Macro Assembler (masm) ile derlemesi gerektiğini belirtmek için, çözüm gezgini penceresinde oluşturduğunuz dosya üzerinde sağ tıklayarak **Derleme Bağımlılıkları** -> **Özelleştirme Oluştur** seçtiğinizde çıkan listedeki seçili olmayan **masm**'yi seçerek Tamam butonu ile onaylamanız gerekmektedir. Bu aşamadan sonra projenize ekleyeceğiniz asm uzantılı dosyalar, VS tarafından masm (seçtiğiniz hedefe bağlı olarak ml64.exe veya ml.exe) ile derlenecektir.

Gerek duyulan C++ ve Assembly kaynak dosyalar hazır olduğu için VS'da x64 Debug ve Release çalıştırılabilir kodu üretmek için proje derlendiğinde ilgili dizin (x64\Debug veya x64\Release) altında kodları üretilecektir. Üretilen çalıştırılabilir kodu, Başlat menüsünden Komut İstemi açarak ilgili çalıştırılabilir dosyanın olduğu dizine giderek çalıştırabilirsiniz. Bu yönergede verilen C++ kod içindeki 32-bit tamsayı (int) elemanları olan 100 elemanlı dizi'nin tanımlandığı ve Assembly dilindeki örnekte 64-bit uzunluğundaki göstericisi **FastCall** çağırma varsayımı ile RDX

kaydedicisi içinde aktarılan dizinin **yalnız ilk üç** elemanının toplamının 32-bit EAX kaydedicisi içinde geri döndürüldüğüne dikkat ediniz. Bu Assembly dilindeki programı, döngüler kullanarak daha genel ifade edilirse, yalnız ilk üç elemanın toplamı yerine aktarılan dizi içindeki tüm elemanların toplamını hesaplayabilecek bir Assembly dili programa dönüştürülebilir.

Sınıflama için tamsayı özellik vektör elemanları arasındaki farkların mutlak değerini hesaplamak için gerek duyulan *abs* emri, Intel işlemcileri emir kümesinde bulunmadığı için, mutlak değeri alınacak sayının değerini bir kaydedicide geçici olarak saklayıp sayıyı 2'ye tümleyene *neg* reg32 emri ile tümledikten sonra tümlenen sayının 0'dan küçük olması durumunda koşullu aktarım emri (**C**onditionally**MOV**eIf**L**ess, küçükse aktar, S≠O) *cmovl* reg32, reg32 ile sayının tümlenmemiş halini geri alarak mutlak değerini bulabilirsiniz. Aşağıdaki kod parçası **eax** kaydedicisi içindeki sayının değerini önce **ebx**'de saklayarak **eax**'i tümlemekte ve sonuç negatif çıkmış ise koşullu aktarım komutu ile sayının pozitif halini **eax**'e geri yüklemektedir [2].

```
mov   ebx, eax      ; eax'i geçici olarak ebx'e sakla
neg   eax           ; eax'i 2'ye tümle
cmovl eax, ebx      ; sonuç negatif ise pozitif ebx'i eax'e geri yükle
```

Çoğu yüksek seviyeli dil derleyicileri tarafından da günümüzde *int* **abs**(*int*) fonksiyonu için üretilmekte olan aşağıdaki kod parçası [3], daha da hızlı mutlak değer hesaplama için, normal (koşul testi gerçekleştirmeyen, *mov* emri gibi) emirlere göre göreceli olarak daha yavaş çalışan koşul testi ile aktarım yapan (yukarıdaki örnekteki cmovl emri gibi) emirleri kullanmadan aşağıdaki kod parçası ile tamsayı mutlak değer hesaplamaktadır.

```
cdq                 ; eax'in işaretini edx'e sakla, + ise 0, - ise 0xFFFFFFFF
xor   eax, edx      ; işareti - ise 1'e tümle, + ise değişmez
sub   eax, edx      ; işareti - ise 2'ye tümleyen için 1 ekle (-(-1)), + ise değişmez
```

Sınıflama ödevinin gelişitirilmesinde yaralanabileceğiniz bu örnek C++ konsol uygulaması gerekli değişiklikleri yapılarak, giriş dosyasından okuduğu bir örneğe ait tamsayılardan oluşan özellik vektörünün, karakteristik özellik vektörleri bilinen A ve B gibi iki farklı sınıftan hangisine ait (daha yakın) olduğunu belirleyerek geri döndüren Assembly dilinde yazılmış bir fonksiyonu çağırarak, fonksiyonunu geri dönüş değerine bağlı olarak örneğin hangi sınıfa ait olduğunu ekrana basmak için kullanılabilir.

Kendisine aktarılan argumanları kullanarak iki sınıf için sınıflama yapan ve sınıf bilgisini geri döndüren örnek bir Assembly kodunu inceleyerek çalıştırabilirsiniz.

Kaynaklar :

1. YouTube'da, “What's a Creel” kullanıcısının “Modern x64 Assembly” eğitim videoları serisi

2. https://stackoverflow.com/questions/2639173/x86-assembly-abs-implementation

3. https://helloacm.com/optimized-abs-function-in-assembly/