فصل 1

C# و .NET Framework

* قبل از .NET
* ورود Microsoft.net
* کامپایل به زبان میانی
* کامپایل به کد محلی و اجرایی
* هسته اجرایی زبان مشترک
* زیر ساخت های زبان مشترک
* مروری بر کلمات اختصاری
* روند تکامل C#

قبل از دات نت

زبان برنامه نویسی C# برای توسعه برنامه های مایکروسافت دات نت فریم ورک طراحی شده بود. این فصل نگاه مختصری بر دات نت دارد، که از کجا آمده و معماری اصلی آن چیست. برای شروع اجازه دهید که نام درست C# را بفهمیم : C# به صورت " سی شارپ[[1]](#footnote-1) " تلفظ می شود.

برنامه نویسی ویندوز در اواخر دهه 1990

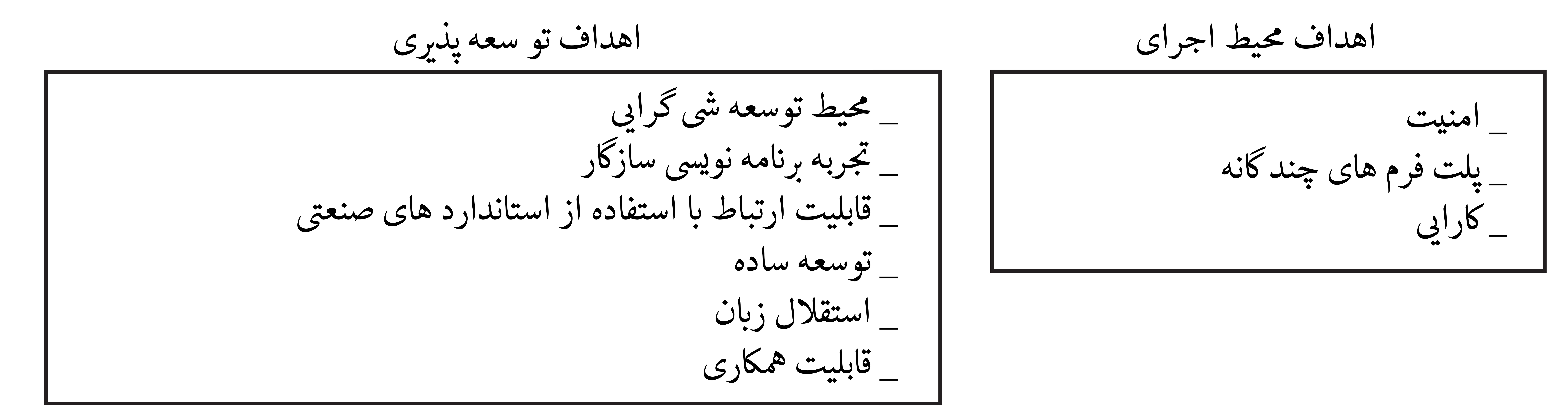
در اواخر دهه 1990 برنامه نویسی ویندوز به تعدادی از شاخه ها شکسته شد. اکثر برنامه نویسان از ویژوال بیسیک (VB[[2]](#footnote-2)) و C و C++ استفاده می کردند. برخی از برنامه نویسان C و C++ از APIهای خام Win32 استفاده می کردند، اما اغلب از کلاس های بنیادین مایکروسافت (MFC[[3]](#footnote-3)) استفاده می کردند. دیگر برنامه نویسان به مدل شیء کامپوننت (COM[[4]](#footnote-4)) نقل مکان کردند.

همه این تکنولوژی ها مشکلات خاص خود را داشتند. APIهای خام Win32 شی گرا نبودند و استفاده از آنها نیاز به کار بیشتری نسبت به MFC داشتند. MFC شی گرا بود ولی در قسمت هایی همخوانی نداشت و همچنین قدیمی شده بود. COM گرچه مفهومی ساده داشت اما در برنامه نویسی واقعی پیچیده بود و ساختار ناهنجار و زشتی را به وجود می آورد.

یکی دیگر از ضعف های برنامه نویسی این فناوری ها این بود که آنها در درجه اول توسعه کد را برای دسکتاب مورد هدف قرار داده بودند و کم تر به توسعه کد های تحت وب توجه می کردند. در آن زمان، تازه برای برنامه نویسی تحت وب چاره اندیشی شده بود و از برنامه نویسی دسکتاب بسیار متفاوت تر به نظر می رسید.

اهدافی برای ارائه خدمات نسل بعدی پلت فرم ها

چیزی که ما واقعاً به آن نیاز داشتیم یکپارچگی و چارچوبی جامع برای شی گرایی بود که انسجام و ظرافت را برای برنامه نویسی به ارمغان بیاورد. مایکروسافت برای پاسخگویی به این نیاز، یک محیط اجرایی و یک محیط توسعه پذیر تنظیم کرد که این اهداف را عملی کند. این اهداف در شکل 1-1 لیست شده است.



شکل 1-1. اهدافی برای نسل بعدی پلت فرم ها

ورود Microsoft.net

در سال 2002 مایکروسافت اولین نسخه از .NET Framework را منتشر کرد و وعده داد که به مشکلات گذشته رسیدگی می کند و پاسخگوی اهداف سیستم های نسل بعدی است. .NET Framework خیلی پایدارتر و شی گرا تر از هر یک از تکنولوژی های برنامه نویسی MFC و COM است. برخی از ویژگی های آن عبارت است از:

* پلت فرم های چند گانه: این سیستم بر روی طیف گسترده ای از کامپیوتر ها، سرور ها و ماشین های رومیزی گرفته تا PADها و تلفن های همراه قابل اجرا است.
* استاندارد های صنعتی: این سیستم از پروتکل های ارتباطی استاندارد صنعتی از قبیل XML ، HTTP ، SOAP و WSDL استفاده می کند.
* امنیت: این سیستم می تواند محیط اجرایی بسیار امن تر را، حتی در حضور کد های بدست آمده از منابع مشکوک فراهم کند.

اجزای .NET Framework

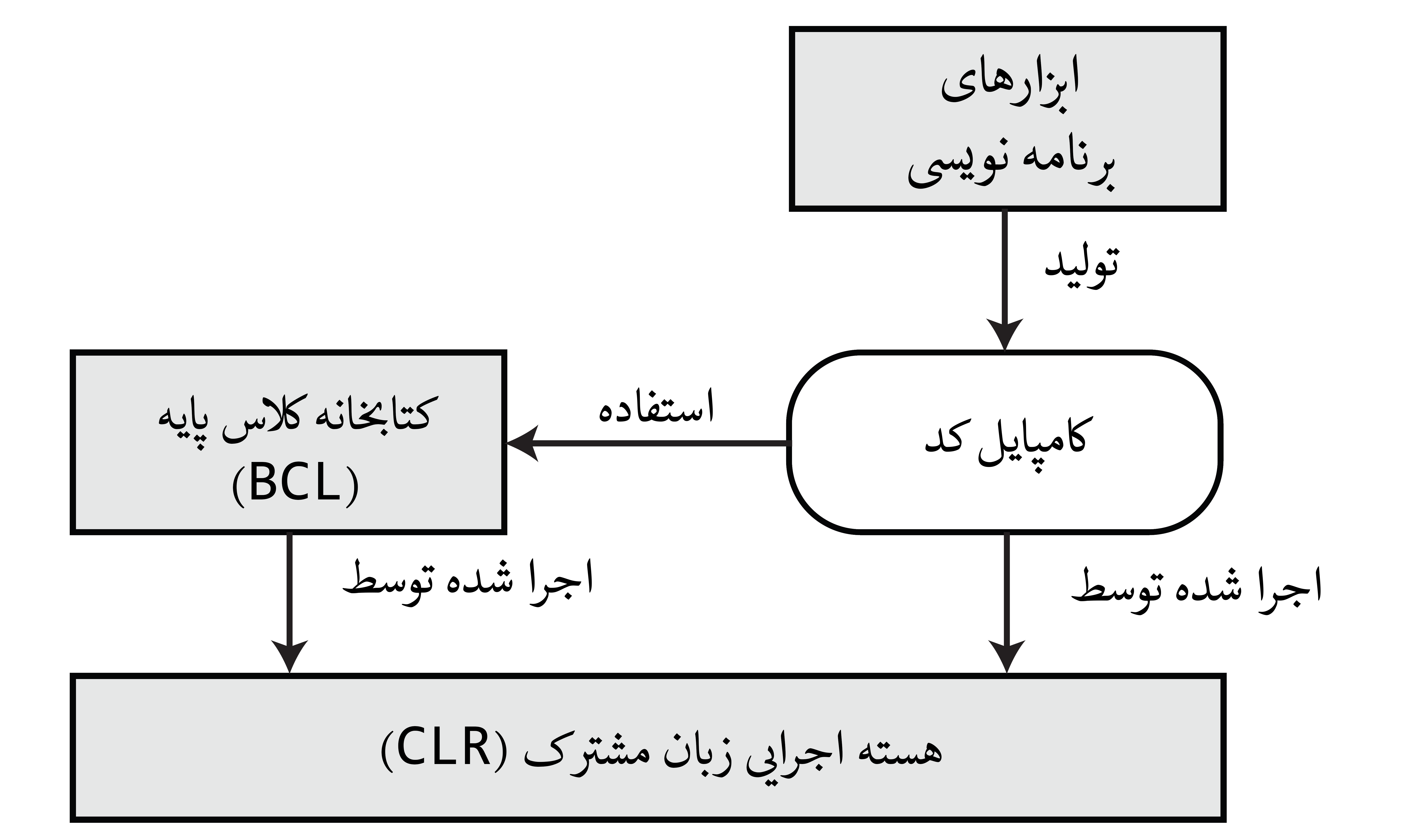
چار چوب دات نت از سه بخش نشان داده شده در شکل 2-1، تشکیل شده است. محیط اجرایی را هسته اجرایی زبان مشترک (CLR) می نامند. CLR برنامه را در زمان اجرا مدیریت می کند، که شامل موارد زیر است:

* مدیریت حافظه و جمع آوری زباله[[5]](#footnote-5)
* تایید ایمن بودن کدها
* اجرای کد، مدیریت محتوا و رسیدگی به موارد استثنا

ابزار های برنامه نویسی شامل مواردی است که برای کد نویسی و اشکال زدایی برنامه نیاز است، ابزار های برنامه نویسی شامل موارد زیر است:

* محیط توسعه یکپارچه (IDE[[6]](#footnote-6)) ویژوال استودیو[[7]](#footnote-7)
* کامپایلر های سازگار با دات نت ( به عنوان مثال C# ، VB.NET ، Jscript ، F# ، IronRuby و مدیریت C++ )
* ابزار های اشکال زدایی[[8]](#footnote-8)
* توسعه فن آوری های تحت شبکه از قبیل ASP.NET یا WCF .

کتابخانه کلاس پایه (BCL) یک کتابخانه کلاسی بزرگ است که در چارچوب دات نت مورد استفاده قرار می گیرد و شما قادر هستید که آن را به بهترین شکل در برنامه هایتان مورد استفاده قرار دهید.



شکل 2-1. اجزای .NET Framework

پیشرفت محیط برنامه نویسی

چارچوب دات نت پیشرفت های قابل توجهی را نسبت به محیط های برنامه نویسی گذشته به برنامه نویسان ارائه می دهد. بخش زیر خلاصه ای از ویژگی ها و فواید آن را ارائه می دهد.

محیط توسعه شی گرایی

BCL ، CLR و C# طراحی شده اند تا کاملاً شی گرا باشند و به عنوان محیطی یکپارچه عمل کنند.

این سیستم برای برنامه های محلی و سیستم های توزیع شده، مدلی شی گرا از برنامه نویسی ارائه می دهد. همچنین این سیستم یک رابط توسعه نرم افزاری برای برنامه نویسی دسکتاپ و برنامه نویسی نرم افزار تلفن همراه و توسعه وب و سازگار با طیف گسترده ای از اهداف، از سرویس دهنده به تلفن های همراه را فراهم می کند.

جمع آوری خودکار زباله

CLR یک سرویس به نام جمع آوری زباله (CG[[9]](#footnote-9)) دارد، که به صورت خودکار حافظه را برای شما مدیریت می کند.

* GC به طور خودکار اشیائی را که برنامه شما دسترسی طولانی مدت به آنها نداشته از حافظه حذف می کند.

GC برنامه نویسان را از کارهایی که به طور رایج انجام می دادند، از قبیل آزاد سازی حافظه و جستجو برای نشست های حافظه راحت می کند. این پیشرفت بزرگی است، چون جستجو برای نشست های حافظه می تواند سخت و وقت گیر باشد.

ارتباط متقابل

چارچوب دات نت برای ارتباط متقابل با دیگر زبان های دات نت، سیستم عامل یا DLL های Win32 و COM طراحی شده است.

* ارتباط متقابل زبان دات نت این اجازه را می دهد تا برنامه ی نوشته شده با استفاده از زبان های مختلف دات نت تعاملی یکپارچه داشته باشند.
  + برنامه ای که در یک زبان دات نت نوشته شده است، تا زمانی که مقررات خاصی را دنبال کند، می تواند از یک کلاس نوشته شده در زبان دیگر دات نت استفاده و حتی ارث بری کند.
  + زبان دات نت به دلیل توانایی یکپارچه سازی مدل های ایجاد شده در زبان های برنامه نویسی، گاهی به عنوان یک زبان اگناستیک[[10]](#footnote-10) توصیف می شود.
* دات نت ویژگی به نام فراخوانی پلت فرم (P/Invoke) را فراهم کرده است که به کد های نوشته شده برای دات نت اجازه می دهد تا از کد هایی که برای دات نت نوشته نشده اند فراخوانی و استفاده کنند. این ویژگی می تواند از توابع خام C که از DLL های استاندارد Win32 وارد شده اند استفاده کند، از قبیل API های ویندوز.
* همچنین چار چوب دات نت اجازه همکاری با COM را دارد. اجزای دات نت می توانند اجزای COM را فراخوانی کنند و اگر اجزای دات نت از اجزای COM باشند، اجزای COM می توانند با اجزای دات نت تماس بگیرند.

عدم نیاز COM

چار چوب دات نت اجازه می هد که برنامه نویس از COM ارث بری کند. اگر شما از محیط برنامه نویسی COM آمده باشید خوشحال خواهید شد که به عنوان یک برنامه نویس C#، احتیاج ندارید که از موارد زیر استفاده کنید:

* واسط **IUnknown**: در COM ، همه اشیا باید از رابط IUknown پیاده سازی شوند. در مقابل همه اشیای دات نت از یک کلاس به نام شی مشتق می شود. هنوز هم واسط برنامه نویسی بخش مهمی از دات نت است اما این دیگر موضوع مرکزی نیست.
* کتابخانه های نوع: در COM ، برنامه نویس مجبور است تا مسیر ارجاعات را نگه داری کند تا در آن زمان مطمئن شود که اشیا به اشتباه حذف نشده باشند. در دات نت، GC مسیری از ارجاعات را نگه داری می کند و تنها در زمان مناسب اشیا را حذف می کند.
* **HRESULT:** COM از نوع داده ای HRESULT استفاده کرده است تا کد های خطا را در زمان اجرا برگرداند. دات نت از HRESULT استفاده نکرده است. در عوض برای همه خطا های غیر منتظره در زمان اجرا، خطای استثنا تولید می کند.
* ریجستری: برنامه های COM باید در ریجستری سیستم ثبت شوند تا اطلاعات مربوط به تنظیمات سیستم عامل و برنامه را نگه داری کند. برنامه های دات نت نیازی به استفاده از ریجستری ندارند. این برنامه ها ساده نصب و یا حذف می شوند. ( با این حال چیزی شبیه به حافظه مونتاژ جهانی[[11]](#footnote-11) نامید می شود که در فصل 21 بررسی خواهد شد.)

اگر چه مقدار کد COM که در حال نوشته شدن است نسبتا کوچک باشد، بازهم تعدادی از اجزای COM در سیستم وجود دارد که مورد استفاده قرار می گیرند، و برنامه نویسان C# گاهی نیاز به کد نویسی دارند که واسطی بین اجزای آنها باشد. C# 4.0 چندین ویژگی جدید معرفی کرده است که کار را آسان می سازد. این ویژگی ها در فصل 25 بررسی خواهند شد.

توسعه ساده

توسعه برنامه های نوشته شده برای دات نت به دلایل زیر می توانند خیلی آسان تر از قبل باشد:

* این واقعیت است که برنامه های دات نت نیازی ندارند که در ریجستری ثبت شوند به این معنی که در راحت ترین حالت یک برنامه فقط نیاز دارد تا در دستگاه مورد نظر کپی شده باشد و آنگاه برای اجرا آماده است.
* دات نت یک ویژگی به نام side-by-side execution[[12]](#footnote-12) ارائه می دهد، که اجازه می دهد تا نسخه های مختلفی از یک DLL در یک دستگاه وجود داشته باشد. این به این معنی است که هر برنامه اجرایی می تواند به نسخه از DLL که برای آن ساخته شده دسترسی داشته باشد.

امنیت نوع

CLR امنیت نوع پارامتر ها و داده ها دیگر اشیا و حتی بین اجزای نوشته شده در زبان های مختلف برنامه نویسی را کنترل و تضمین می کند.

کتابخانه کلاس پایه

چارچوب دات نت کتابخانه بزرگی از کلاس های پایه را فراهم کرده است که کتابخانه کلاس پایه (BCL[[13]](#footnote-13)) نامیده می شود. ( گاهی اوقات کتابخانه کلاس فریم ورک (FCL[[14]](#footnote-14)) هم نامیده می شود.) شما می توانید در هنگام نوشتن برنامه های خود از کد های موجود در این مجموعه بزرگ استفاده کنید. برخی از طبقه بندی های به شرح زیر است:

* کلاس های پایه: کلاس هایی که با مجموعه ای از ابزار های قدرتمند برای طیف وسیعی از وظایف برنامه نویسی از جمله کار با فایل، کار با رشته ، امنیت و رمز نگاری در دسترس شما قرار می گیرند.
* کلاس های مجموعه[[15]](#footnote-15): کلاس هایی که پیاده سازی لیست ها ، لغت نامه ها ، جداول hash و آرایه ها را برعهده دارد.
* کلاس های هماهنگ سازی و Threading: کلاس هایی برای ساخت برنامه های چند پردازشی[[16]](#footnote-16)
* کلاس های XML: کلاس هایی برای ایجاد، خواندن و کار با اسناد XML

کامپایل به زبان میانی

کامپایلر زبان دات نت فایل کد منبع را می گیرد و فایل خروجی را تولید می کند که اسمبلی[[17]](#footnote-17) نامیده می شود. مراحل کامپایل در شکل 3-1 نشان داده شده است.

* اسمبلی یک فایل اجرایی و یا یک فایل DLL است.
* کد درون یک اسمبلی کد محلی[[18]](#footnote-18) ماشین نیست بلکه یک زبان میانی (CIL[[19]](#footnote-19)) نامیده می شود.
* یک اسمبلی موارد زیر را شامل می شود:
  + برنامه های CIL
  + فرا داده مربوط به نوع های استفاده شده در برنامه
  + فرا داده مربوط به ارجاع دیگر اسمبلی های



شکل 3-1. فرآیند کامپایل

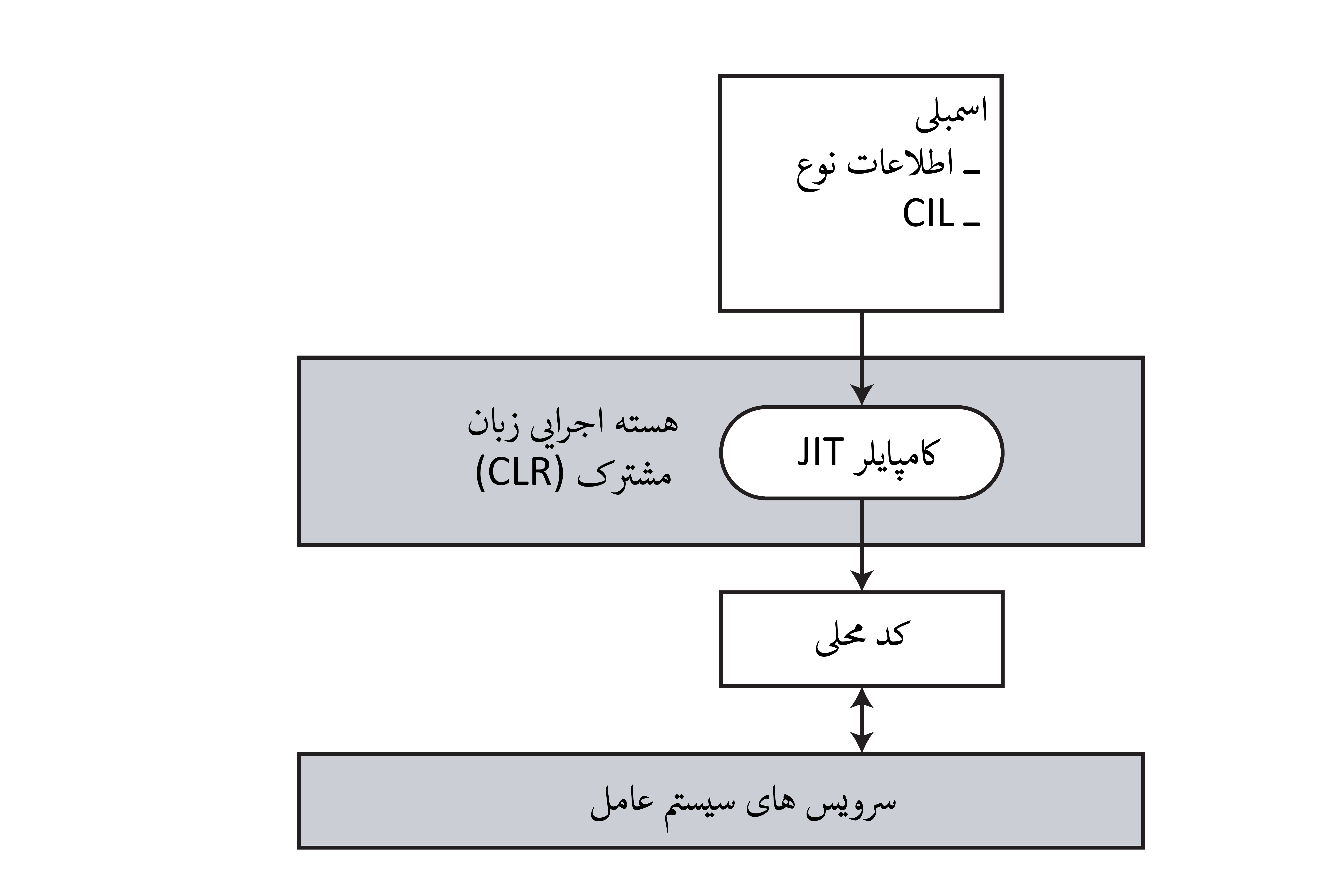
|  |
| --- |
| * توجه مخفف زبان میانی در طول زمان تغییر کرده است و منابع مختلف از اصطلاحات مختلفی استفاده کرده اند. دو اصطلاح دیگر برای CL به کار می رود که عبارت است از Intermediate Language[[20]](#footnote-20) (IL) و Microsoft Intermediate Language[[21]](#footnote-21) (MSIL) . این اصطلاحات اغلب در طی توسعه اولیه و اسناد قدیمی دات نت استفاده شده اند، اما باز هم از آنها استفاده می شوند. |

کامپایل به کد محلی و اجرایی

برنامه های CIL تا زمانی که اجرا نشوند به کد محلی ماشین تبدیل نمی شود. در زمان اجرا، CLR همانطور که در شکل 4-1 نشان داده شده است مراحل زیر را انجام می دهد:

* ویژگی های امنیتی اسمبلی آن را بررسی می کند.
* فضایی در حافظه به آن اختصاص می دهد.
* کد های اجرایی اسمبلی فقط در این زمان (JIT) به کامپایلر ارسال می شود تا بخش هایی از آن را به کد محلی تبدیل کند.

در اسمبلی تنها کد هایی که به آنها نیاز است به وسیله کامپایلر JIT کامپایل می شوند. سپس آن را در مکانی ذخیره می کند که برای اجرای دوباره این برنامه آنها را مورد استفاده قرار دهد. این فرایند به این معنی است که کد هایی که در زمان اجرا فراخوانی نشده اند به کد های محلی کامپایل نشده اند و فقط کد هایی که فراخوانی شده اند یکبار کامپایل شده اند.



شکل 4-1. کامپایل به کد محلی در زمان اجرا

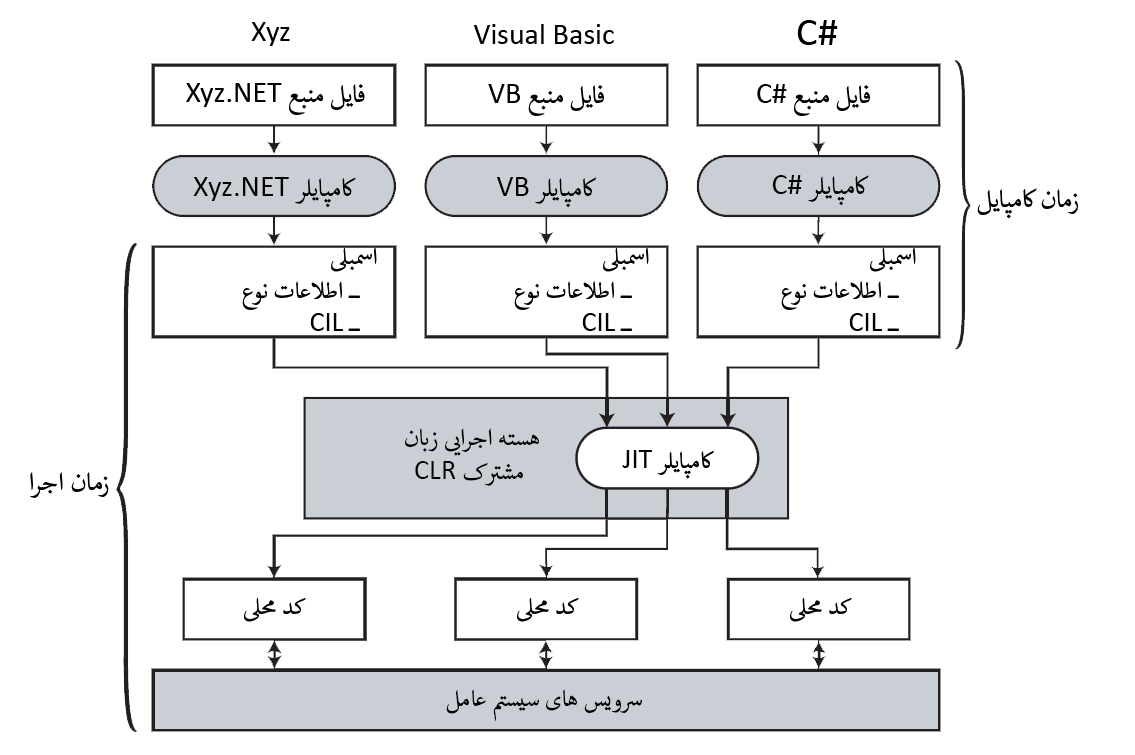
زمانی که CIL به کد محلی تبدیل می شود، CLR آن را در زمان اجرا مدیریت می کند، CLR وظایفی مانند آزاد سازی حافظه بدون استفاده، بررسی محدوده آرایه ، بررسی مقادیر نوع ها، مدیریت استثنائات و... را انجام می دهد. این موضوع دو اصطلاح مهم را به ارمغان می آورد:

* کد مدیریت شده[[22]](#footnote-22): کد نوشته شده برای دات نت فریم ورک، کد مدیریت شده نامیده می شود و به CLR نیاز دارد.
* کد مدیریت نشده[[23]](#footnote-23): کدی که زیر نظر CLR اجرا نمی شود، از جمله DLLهای C/C++ ، Win32 که کد های مدیریت نشده نامیده می شود.

همچنین مایکروسافت ابزاری به نام Native Image Generator ، یا Ngen فراهم کرده است که یک اسمبلی دریافت می کند و کد محلی را برای فرایند جاری تولید می کند. کدی که از طریق Ngen اجرا می شود در زمان اجرا از فرایند کامپایل JIT اجتناب می کند.

مروری بر کامپایل و اجرا

فرایند کامپایل و اجرا صرفنظر از زبان برنامه، از فایل های منبع پیروی می کند. شکل 5-1 کامپایل نهایی و فرایند اجرای هر سه برنامه نوشته شده با زبان های مختلف را نشان می دهد.

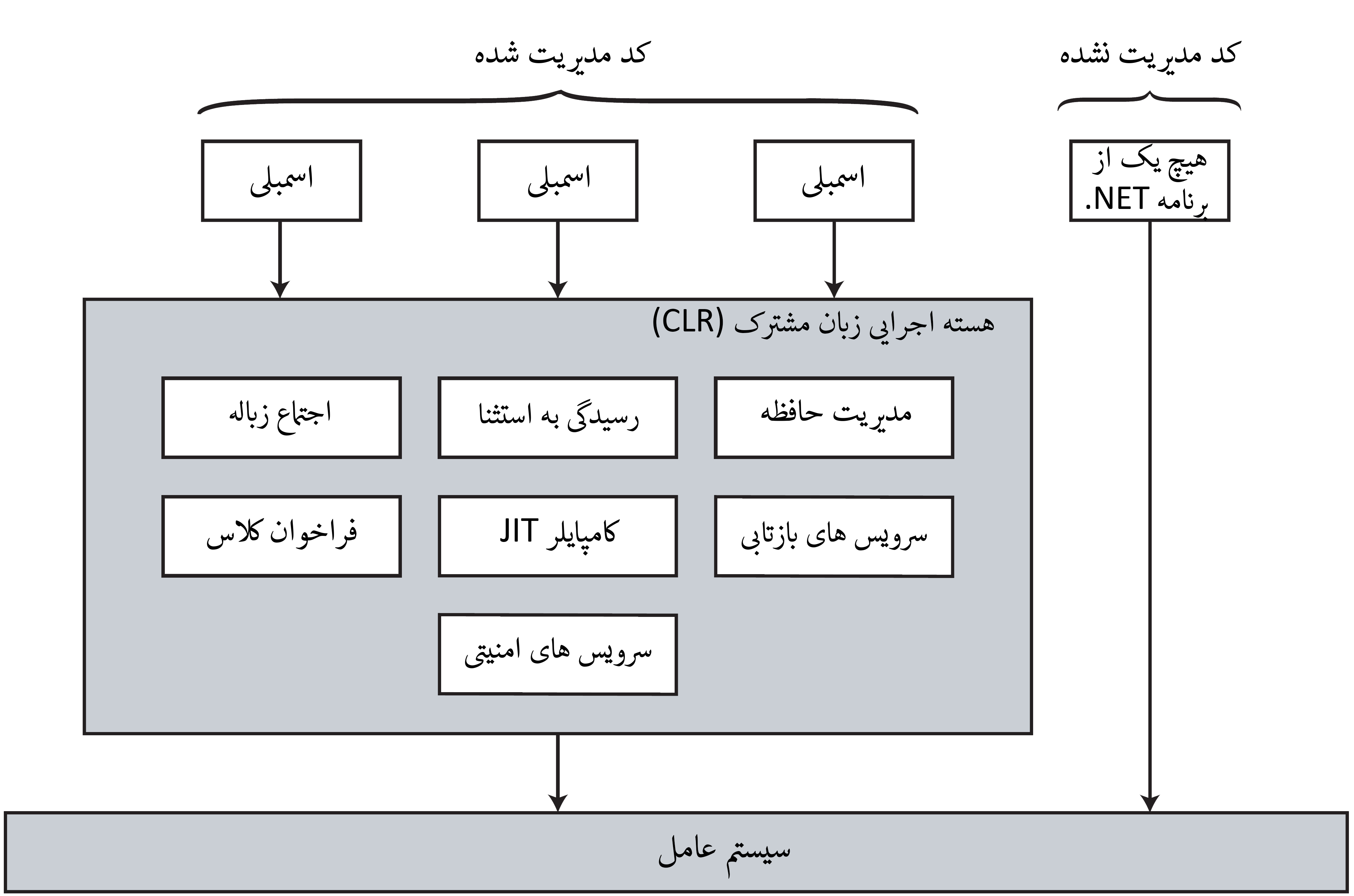


شکل 5-1. مروری بر کامپایل و فرایند اجرا

هسته اجرایی زبان مشترک CLR

CLR[[24]](#footnote-24) جزء هسته اصلی دات نت است، و همانطور که در شکل 6-1 نشان داده شده است CLR در راس سیستم عامل و مدیریت برنامه های اجرایی قرار دارد، همچنین CLR خدمات زیر را ارائه می دهد:

* جمع آوری خودکار زباله
* امنیت و تأیید هویت
* قابلیت برنامه ریزی گسترده از طریق دسترسی به BCL – شامل قابلیت هایی نظیر سرویس های وب و سرویس های داده



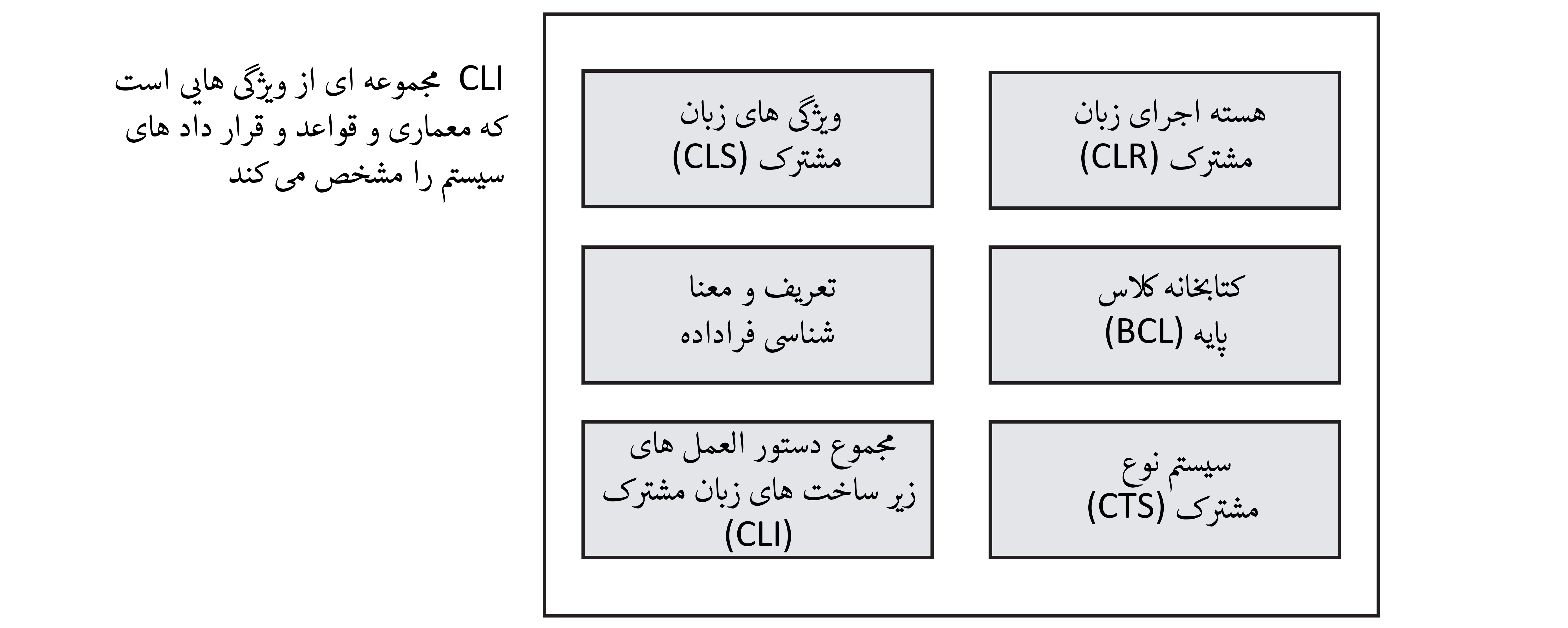
شکل 6-1. مروری بر CLR

زیر ساخت های زبان مشترک

هر زبان برنامه نویسی دارای مجموعه ای از نوع های اصلی است اشیایی از قبیل Integer[[25]](#footnote-25) ، Float[[26]](#footnote-26) ، Character[[27]](#footnote-27) و غیره. از لحاظ تاریخی، ویژگی های این نوع ها از یک زبان برنامه نویسی به زبان های برنامه نویسی دیگر و از یک پلت فرم به پلت فرم دیگر متفاوت است. برای مثال، تعداد بیت های تشکیل دهنده یک عدد صحیح بسته به گستردگی زبان و پلت فرم متفاوت است.

این عدم یک نواختی، زمانی که بخواهیم برنامه ای را با دیگر برنامه ها و کتابخانه های نوشته شده در دیگر زبان ها ارتباط دهیم و به خوبی اجرا شود مشکل ساز می شود. برای داشتن نظم و همکاری، باید مجموعه ای از استاندارد ها وجود داشته باشد.

زیر ساخت های زبان مشترک (CLI[[28]](#footnote-28)) مجموعه ای از استاندارد ها است که روابط تمام اجزای دات نت را در سیستم منسجم و استوار می سازد. CLI مفاهیم و معماری سیستم و قواعد و قرار داد هایی که تمام نرم افزار ها باید از آنها پیروی کنند را مشخص می سازد. شکل 7-1 اجزای CLI را نشان می دهد.



شکل 7-1. اجزای CLI

ویژگی های استاندارد بین المللی CLI و C# توسط ECMA[[29]](#footnote-29) International تصویب شده اند. ( نام ECMA برای انجمن سازندگان کامپیوتر اروپا استفاده می شود اما در حال حاضر فقط یک کلمه را در خود دارد.) اعضای ECMA شامل مایکروسافت[[30]](#footnote-30) ، IBM، هیولت پاکارد[[31]](#footnote-31) ، ادوب[[32]](#footnote-32) ، و بسیاری از شرکت های مرتبط با کامپیوتر و مصرف کنندگان الکترونیک است.

بخش های مهمی از CLI

اگر چه بسیاری از برنامه نویسان نیازی به دانستن جزئیات CLI ندارند، ولی حداقل باید با معنی و هدف سیستم نوع مشترک و زبان مشترک آشنا باشند.

سیستم نوع مشترک

سیستم نوع مشترک (CTS[[33]](#footnote-33)) ویژگی نوع هایی که در کد مدیریت شده استفاده می شود را تعریف می کند. برخی از جنبه های مهم CTS به شرح زیر است:

* CTS مجموعه ای از نوع های و ویژگی های خاص را برای هر نوع، تعریف می کند.
* نوع هایی که توسط زبان برنامه نویسی سازگار با دات نت ارائه شده است به طور کلی به برخی از زیر مجموعه های خاص از مجموعه نوع های اصلی تقسیم بندی می شود.
* یکی از مهم ترین ویژگی های CTS این است که همه نوع ها از یک کلاس پایه مشترک به نام شی[[34]](#footnote-34) مشتق شده است.
* استفاده از CTS تضمین می کند که نوع های سیستم و نوع های تعریف شده توسط کاربر می توانند توسط زبان های .NET استفاده شوند.

مشخصات زبان مشترک

مشخصات زبان مشترک (CLS[[35]](#footnote-35)) مشخص کننده قوانین، خصوصیات و رفتارهای زبان برنامه نویسی سازگار با دات نت می باشد. این موضوعات شامل نوع های داده ، سازنده کلاس و ارسال پارامتر می باشد.

مروری بر کلمات اختصاری

این فصل بسیاری از کلمات اختصاری دات نت را بیان می کرد، پس شکل 8-1 به شما کمک خواهد کرد که این اصطلاحات را راحت تر به خاطر بسپارید.



شکل 8-1. کلمات اختصاری دات نت

روند تکامل C#

نسخه فعلی این زبان نسخه 5.0 است. به هر نسخه جدید از این زبان ویژگی هایی افزوده شده است. ویژگی جدیدی نسخه 5.0 برنامه نویسی نا همگام[[36]](#footnote-36) است. این ویژگی های گسترده در فصل 20 بررسی می شوند.

شکل 9-1 ویژگی های اصلی هر نسخه را نشان می دهد.



شکل 9-1. ویژگی های منتشر شده از نسخه های C#

1. C Sharp [↑](#footnote-ref-1)
2. Visual Basic [↑](#footnote-ref-2)
3. Microsoft Foundation Class [↑](#footnote-ref-3)
4. Component Object Model [↑](#footnote-ref-4)
5. Garbage collection [↑](#footnote-ref-5)
6. Integrated development environment [↑](#footnote-ref-6)
7. Visual Studio [↑](#footnote-ref-7)
8. Debugger [↑](#footnote-ref-8)
9. Garbage Collection [↑](#footnote-ref-9)
10. Language Agnostic [↑](#footnote-ref-10)
11. Global assembly cache [↑](#footnote-ref-11)
12. اجرای پهلو به پهلو [↑](#footnote-ref-12)
13. Base Class Library [↑](#footnote-ref-13)
14. Framework Class Library [↑](#footnote-ref-14)
15. Collection classes [↑](#footnote-ref-15)
16. برنامه هایی که توانایی مدیریت چندین پروسه به صورت همزمان را دارند. [↑](#footnote-ref-16)
17. Assembly [↑](#footnote-ref-17)
18. Native Code [↑](#footnote-ref-18)
19. Common Intermediate Language [↑](#footnote-ref-19)
20. زبان میانی [↑](#footnote-ref-20)
21. زبان میانی مایکروسافت [↑](#footnote-ref-21)
22. Managed code [↑](#footnote-ref-22)
23. Unmanaged code [↑](#footnote-ref-23)
24. Common Language Runtime [↑](#footnote-ref-24)
25. عدد صحیح [↑](#footnote-ref-25)
26. عدد اعشاری [↑](#footnote-ref-26)
27. کاراکتر [↑](#footnote-ref-27)
28. Common Language Infrastructure [↑](#footnote-ref-28)
29. European Computer Manufacturers Association [↑](#footnote-ref-29)
30. Microsoft [↑](#footnote-ref-30)
31. Hewlett-Packard [↑](#footnote-ref-31)
32. Adobe [↑](#footnote-ref-32)
33. Common Type System [↑](#footnote-ref-33)
34. Object [↑](#footnote-ref-34)
35. Common Language Specification [↑](#footnote-ref-35)
36. Asynchronous programming [↑](#footnote-ref-36)