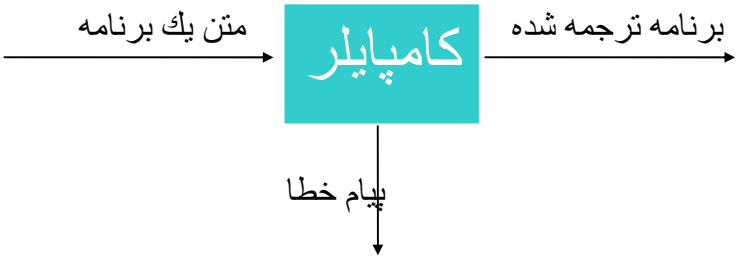
فصل اول

مقدمه

۱.۱ تعريف كامپايلر

كامپايلر برنامه اي است كه در ورودي خود، متن يك برنامه را كه طبق قوانين ودستور زبان تدوين شده براي آن كامپايلر مشخص شده است پذيرفته و در خروجي برنامه اي به زبان دوم ايجاد مي نمايد. اين زبان دوم مي تواند ماشين يا برنامه به زبان ديگر باشد.



شكل 1- ورودي و خروجي كامپايلر

برنامه ترجمه شده در بعضي از محيط ها مثل Unix معمولا" به زبان C ميباشد . براي مثال كامپايلر راتفورد كه نوع پيشرفته تر از فورترن ميباشد در خروجي خود برنامه به زبان C را ايجاد مي نمايد كه توسط كامپايلر C تبديل به كد ماشين مي گردد.

بعضي از كامپايلر ها كه در اصطلاح *مفسر* يا Interpreter ناميده مي شوند همگام با ترجمه هر جمله اجرايي ، آن را به اجرا در مي آورند. براي نمونه زبان نرم افزار Dbase را ميتوان نام برد. البته اين ترجمه همراه با اجرا، زمان اجرايي را بسيار طولاني ميكند، اما امكاناتي در اختيار برنامه نويس قرار ميدهد كه كامپايلرهاي عادي قادر به حصول آن نيستند . براي مثال در زبان Dbase دستورالعمل ماكرو كه با عملگر & مشخص مي شود اين امكان را فراهم ميكند كه بتوان در زمان اجرا محتوي يك رشته را به اجرادرآورد . به مثال زير توجه نمائيد:

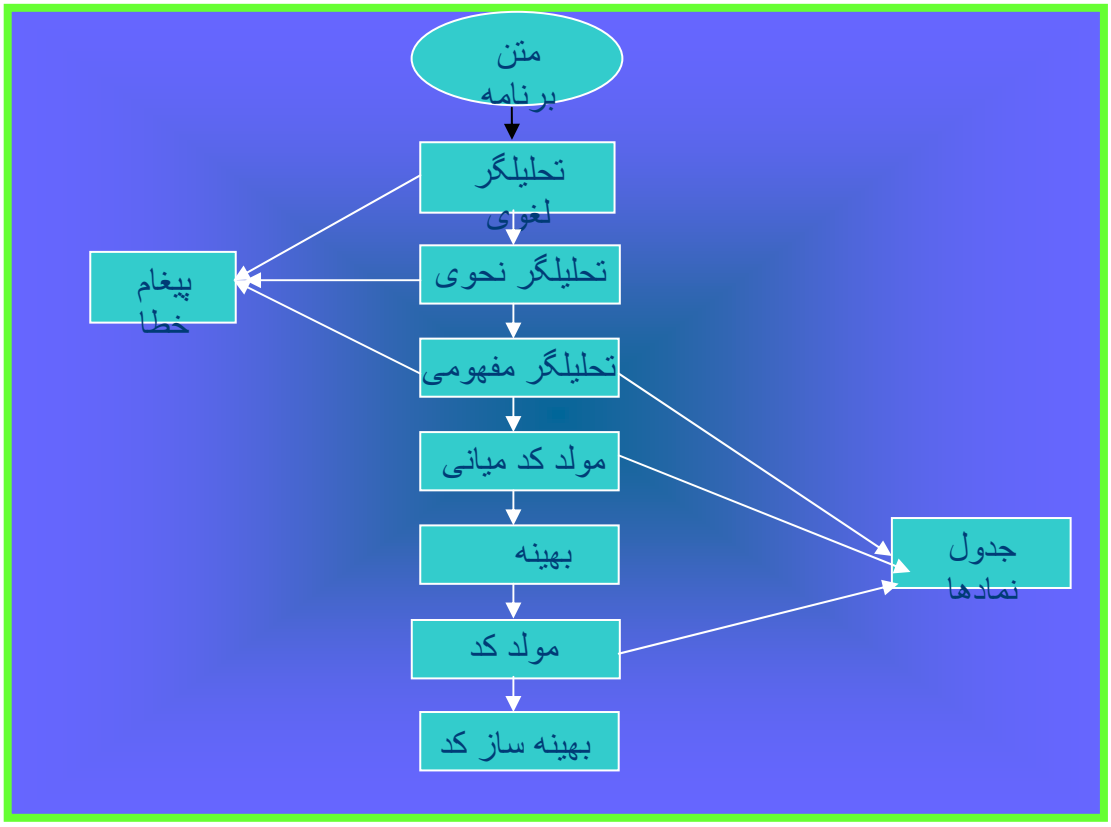
```
A = 'B = 10'
&A
```

```
@ 2,3 get A
&A
```

در مثال فوق با اجراي اولين دستورالعمل &A محتواي A بعنوان يك دستورالعمل تلقي مي شود وبه اجرا در مي آيد و به اين ترتيب محتواي B برابر 10 قرار ميگيرد . عمل كامپايل در طي مراحل مشخص كه در بخش بعدي در مورد آن صحبت خواهد شد انجام ميگردد. اين مراحل ميتوانند بطور همزمان اجرا شوند.

۱.۲ مراحل كامپايلر

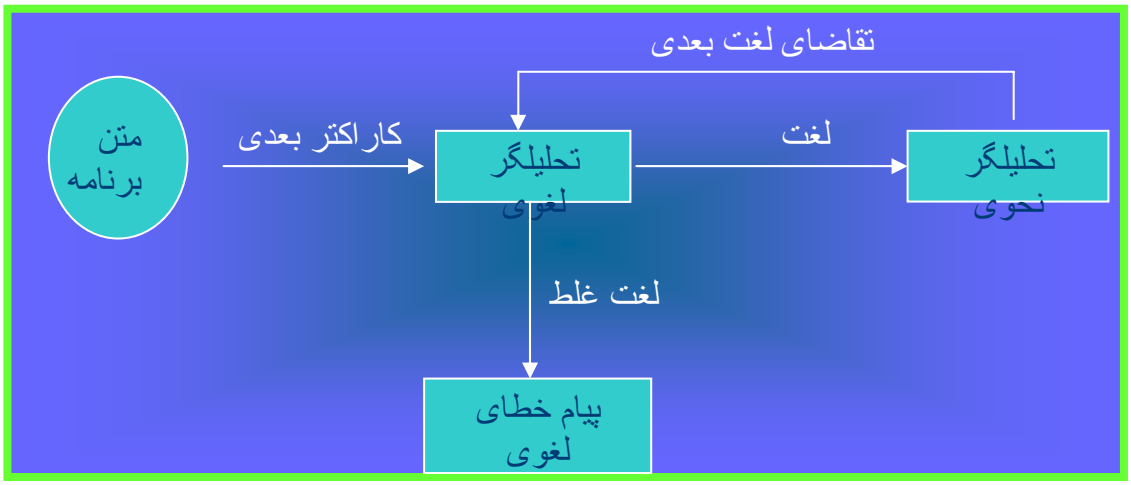
همانطور كه در شكل 1.2 مشخص شده است، كامپايلرها در حالت كلي از هفت بخش اصلي تشكيل مي شوند . در اين بخش بطور خلاصه بخشهاي متفاوت كامپايلر تشريح ميشوند.



شكل 1.2- ساختار كلي كامپايلرها

الف ـ تحليلگر لغوي

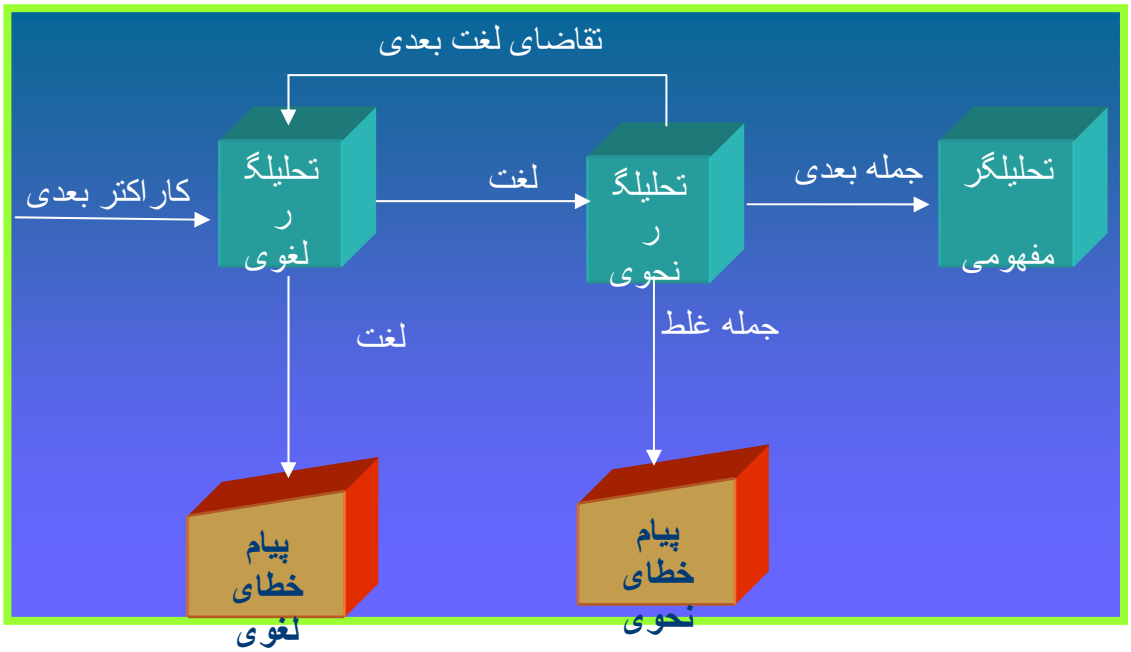
وظيفه تابع ***تحليگر لغوي*** يا در اصطلاح خارجي Lexical Analyser ، تشخيص لغات در متن برنامه مورد كامپايل است كه توسط بخش ديگري از كامپايلر تحت عنوان تحليلگر نحوي مورد فراخواني قرار ميگيرد. با هر بار فراخواني اين لغت بعدي را از متن برنامه تشخيص ميدهد و اطلاعات لازم در مورد لغت را براي تحليلگر نحوي ارسال ميدارد. هر زبان برنامه سازي قوانين لغوي مربوط بخود را دارد . قوانين لغوي بيانگر فرم كلي انواع لغات در زبان برنامه سازي است . تحليلگر لغوي در صورت وجود خطا در قالب بندي لغات استفاده شده در متن برنامه مورد كامپايل اعلام *خطا لغوي* مينمايد. تابع تحليلگر لغوي در فصل دوم از اين كتاب مورد بررسي واقع خواهد شد.



شكل 1.3- شماي تحليلگر لغوي

ب - تحليلگر نحوي

وظيفه ***تحليلگر نحوي*** يا در اصطلاح Syntax Analyser ، تشخيص صحت فرم ظاهري برنامه ها از لحاظ دستورالعمل زبان برنامه سازي مربوطه است . تحليلگر نحوي با فراخواني تحليلگر لغوي لغات را از متن برنامه مورد كامپايل دريافت و صحت قرار گرفتن آنها را در مجاور يكديگر بر اساس دستورالعمل زبان مربوطه مورد آزمون و تحليل نحوي قرار ميدهد. در صورتيكه از لحاظ دستورالعمل زبان ، برنامه مورد كامپايل داراي خطا باشد پيام خطا توسط تحليلگر نحوي صادر ميگردد .

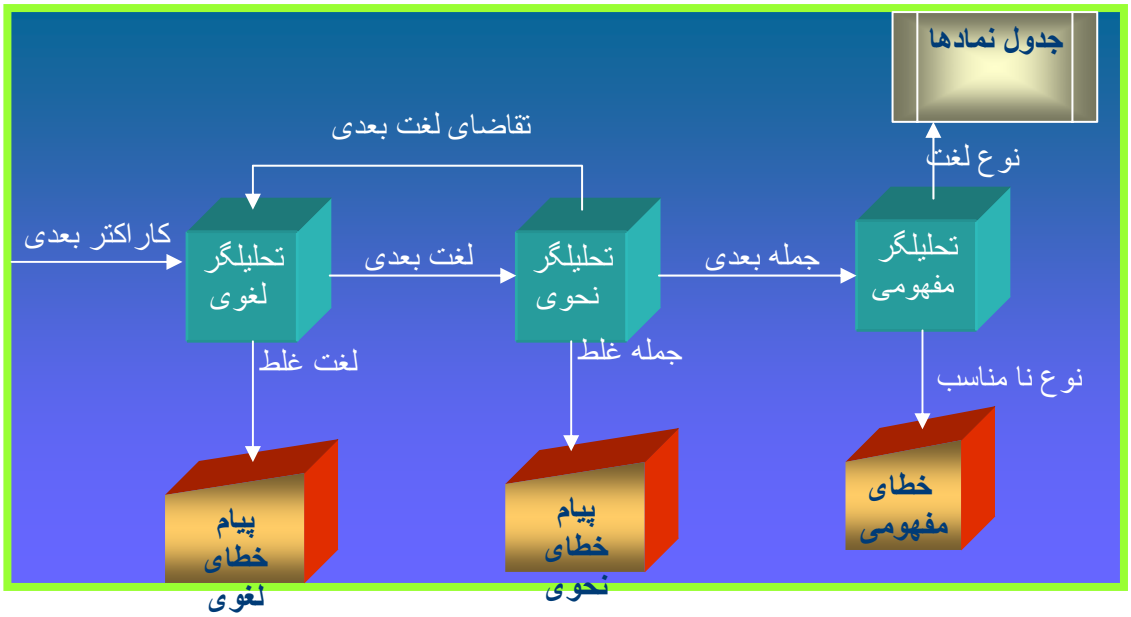


شكل 1.4- شماي كلي تحليلگر نحوي

انواع روشهاي تحليل نحوي و تشخيص صحت برنامه ها بر اساس دستورالعمل و گرامر زبانها برنامه سازي در فصل هاي 3 ، 4 و 5 مورد بحث و بررسي واقع خواهد شد .

ج- تحليلگر مفهومي

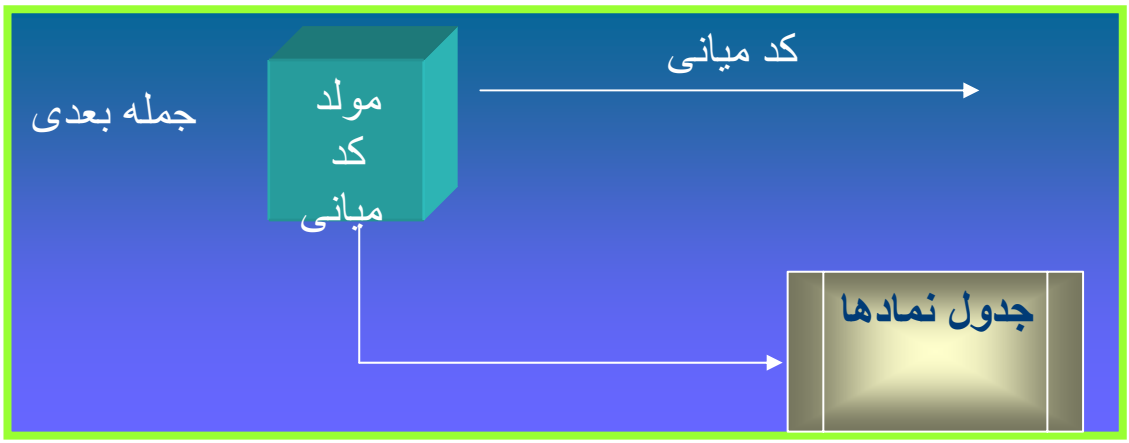
كار تحليلگر مفهومي يا در اصطلاح Semantics Analyser تعيين صحت مفهوم جملات است. ممكن است يك جمله ازنظر نحوي صحيح ولي از لحاظ مفهومي داراي خطا باشد. براي مثال جمله مهردادآتش خورد عليرغم اينكه در دستور زبان فارسي از نظر نحوي صحيح ميباشد از نظر مفهومي نادرست است . در كامپايلرها معمولا" عمل تحليل مفهومي محدود به آزمون نوع يا Type Checking است . تحليلگر مفهومي وابسته به نوع اسامي و متغيرها كه در جدول نمادها مشخص شده، صحت استفاده آنها را جملات و عبارات مختلف مورد آزمون قرار ميدهد. براي مثال اگر متغيري از نوع رشته تعريف شده باشد نمي توان آن را با متغيري از نوع صحيح يا اعشاري جمع و يا مقايسه نمود. فصل 7 در ارتباط با آزمون نوع است.



شكل 1.5- شماي تحليلگر مفهومي

د- مولد كد مياني

مولد كد مياني بخشي از كامپايلر است كه در ورودي خود جملات تشخيص داده شده توسط تحليلگر نحوي را پذيرفته و در خروجي كد واسطه يا مياني توليد ميكند. كد مياني بسادگي قابل تبديل و نزديك به زبان ماشين است اما مستقل از ساختار و جزئيات هر گونه ماشين خاص ميباشد .



شكل 1.6- شماي كلي مولد كد مياني

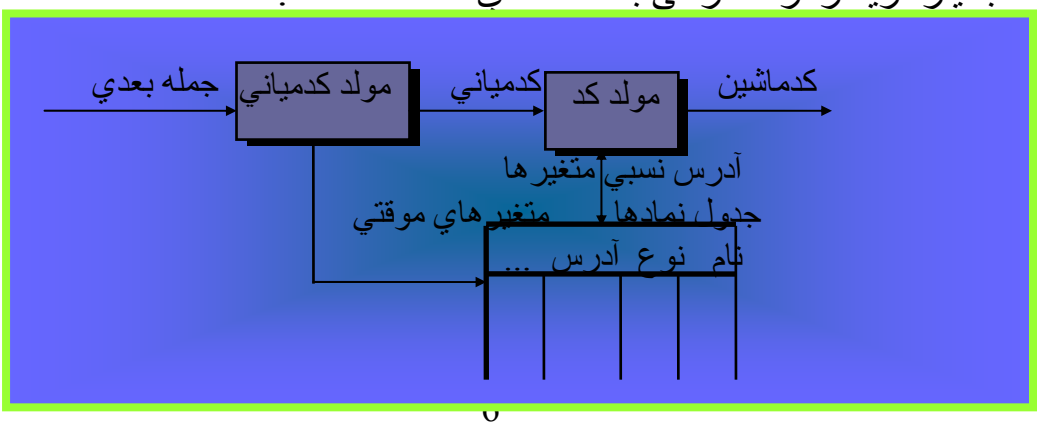
يكي از مزيتهاي ايجاد كد مياني افزايش قابليت حمل كامپايلر بر روي سخت افزار ها با كد متفاوت و ساختار متفاوت است . به اين ترتيب كه برنامه كامپايلر را بتوان بروي كامپيوترها با كد ماشين و اسمبلي متفاوت كامپايل نموده ، با اجراي آن متن برنامه هاي مورد كامپايل را به كد مياني تبديل كرد و سپس با نوشتن يك برنامه كوچك اين كد واسطه را به كد ماشين مورد نظر تبديل نمود. مزيت ديگر ايجاد كد مياني ، فراهم نمودن شرايط خوب براي بهينه سازي كد برنامه ها است. كد مياني در فصل 6 مطرح شده است.

ه- بهينه ساز كد مياني

هدف از بهينه سازي كد مياني ميتواند تقليل حجم و افزايش سرعت اجرايي كدماشين حاصل از كار كامپايلر باشد . براي اين منظور بهينه ساز ، كد مياني حاصل از مولد كد مياني را مورد تحليل قرار مي دهد. معمولا" تحليل كد مياني با آزمون نمادي برنامه ها تحقق مي يابد ، به اين ترتيب كه مفهوم برنامه ها در زمان كامپايل، با تبديل كد مياني به يك گراف بنام گراف جريان مورد بررسي و تحليل واقع شده ، سعي ميكنند تا حجم كد حاصل را تقليل داده ، در صورت وجود كد زائد را مشخص و حذف نمايد. بهينه سازي موضوع فصل 8 از اين كتاب است. تحليل نمادي براي آزمون صحت نرم افزارهاي كاربردي در مباحث مهندسي نرم افزار نيز مطرح ميباشد . مبحث آزمون نرم افزار تحت عنوان Software testing شناخته شده است . تكنيكهاي بهينه سازي در تشخيص توازي كد برنامه ها و اجراي موازي (Parallel) يا هم زمان كد برنامه ها بروي چند پردازنده نيز مورد استفاده قرار ميگيرد .

و- مولد كد

اين بخش از كامپايلر وابسته به ساختار سخت افزار است كه معمولا" فازي جداگانه در كامپايلر ميباشد و كد مياني بهينه شده را تبديل به كد ماشين (Object Code) مينمايد . در الگوريتمهاي توليد كد، معمولا" هدف اينست كه حداكثر بجاي ارجاع به حافظه از ثباتهاي ماشين استفاده شود، چرا كه دسترسي به يك ثبات بسيارسريعتر از دسترسي به خانه هاي حافظه است .



شكل 2.7- شماي مواد كد

همانگونه كه در شكل فوق مشاهده ميشود در زمان كامپايل، آدرس نسبي متغيير ها و اسامي بكار رفته در داخل برنامه ، درون جدول نمادها ثبت ميگردد . اين آدرس هاي نسبي هميشه نسبت به شروع برنامه و در داخل هر زير برنامه نسبت به آدرس شروع زيربرنامه سنجيده ميشود. آدرس شروع براي هر زيربرنامه و برنامه اصلي در زمان كا مپايل صفر درنظر گرفته ميشود . در هنگام پيوند و قرار دادن برنامه ها در حافظه اصلي كه به منظور اجراي برنامه صورت ميگيرد آدرسهاي شروع محاسبه ميگردند . روشهاي توليد كد در فصل 9 مطرح شده است.

ز - بهينه ساز كد

هدف از بهينه سازي كد ، تقليل حجم اشغالي و تسريع كد اجرايي برنامه است . در اين مرحله براي انجام عمل بهينه سازي و رسيدن به اهداف آن ساختار اسمبلي و كد ماشين و امكانات سخت افزاري آن در نظر گرفته ميشود و از دستورالعمل هايي كه سريعتر و با حجم كمتر به اجرا در مي آيند استفاده مي شود بهينه ساز كد در صورت امكان دستورالعمل هاي سريع و كم حجم را با دستورالعمل هاي بكار برده شده در خروجي مولد كد، كه همان كد اجراي برنامه است جايگزين ميكند .

البته مراحل فوق الذكر بعضا" ممكن است در ايجاد كامپايلر حذف شود . براي مثال قسمت بهينه سازي و يا تحليل مفهومي ممكن است در يك كامپايلر وجود نداشته باشد . و نيز ممكن است كليه مراحل در يك مرحله و به صورت همزمان اجرا شود . معمولا" عمليات كامپايل برنامه ها در دو مرحله ويا در اصطلاح در دو Pass انجام مي شود .

۱.۳ كامپايلر كامپايلرها

كامپايلر كامپايلر در واقع يك مولد كامپايلر ميباشد و يا برنامه اي است كه در ورودي خود قوانين لغوي و گرامر و كدي كه براي هر جمله ، عبارت و عنصري از زبان را كه بايد توسط مولد كد ايجاد شود را در ورودي پذيرفته و در خروجي خود يك كامپايلر ايجاد ميكند .

در طي فصل هاي بعدي خواهيم ديد كه چگونه ميتوان يك مولد تحليل گر لغوي و تحليل گر نحوي را بسادگي توليد نمود . از كامپايلر كامپايلرهاي رايج يكي YACC را ميتوان نام برد. اين مولد كامپايلر قواعد گرامري و كد مورد نظر

براي تبديل و در واقع كامپايل جملات متن برنامه كامپايلر را در ورودي خود دريافت ميكند. خروجي YACC برنامه كامپايلر به زبان C است . عمل تحليل لغوي براي اين كامپايلر كامپايلر توسط يك مولد تحليلگر لغوي بنام LEX انجام ميشود. در اين كتاب نشان داده خواهد شد كه چگونه ميتوان يك تحليل گر لغوي ايجاد نمود كه بسيار ساده تر از LEX بتوان آنرا براي تحليل لغوي مورد استفاده قرار داد.

۱.٤ كامپايلر برنامه ها با لغات فارسي

در زبان فارسي بعلت اينكه عبارات از چپ به راست و جملات از راست به چپ نوشته ميشوند ، نميتوان بسادگي با استفاده از روشهاي جاري كامپايلر زبان برنامه سازي كه لغات در آن به زبان فارسي هستند، ايجاد نمود. اگر دستورالعمل هاي زبان برنامه سازي با استفاده از لغات فارسي از سمت چپ به راست نوشته شوند اين مشكل برطرف خواهد شد.

ميتوان برنامه هاي C يا پاسكال و هر زبان برنامه سازي ديگري را با استفاده از لغات فارسي و به زبان فارسي نوشت و با استفاده از يك تحليلگر لغوي و جدول لغات كليدي فارسي و مشابه آن در انگليسي به زبان اصلي تبديل نمود. پس از اشكالزدايي و تكميل برنامه ها ميتوان بالعكس عمل نموده متن انگليسي را به فارسي تبديل نمود.

۱.٥ دلايل تأليف اين كتاب

زبانهاي برنامه سازي ابزار اصلي براي بهره بري از تسهيلات كامپيوتر و كامپايلرها ابزار توليد زبانهاي برنامه سازي هستند. لذا، كامپايلرها يكي از مباحث اصلي و علم مادر در زمينه نرم افزار است. نرم افزار و تكنولوژي كامپيوتر امروزه كليه علوم را تحت الشعاع خود قرار داده است. لذا، ايجاد و صدور قطعات نرم افزاري مي تواند بعنوان يك منبع درآمد و سرمايه ملي مطرح باشد.
پيشرفت در اين مسير بسادگي امكان پذير است زيرا، توليد قطعات نرم افزاري در سطح بازار جهاني نيازي به پشتوانه قوي تكنولوژيك و صنعتي ندارد.
متأسفانه رقابت جهاني موجب حذف مطالب اصلي و كاربردي از داخل كتب علمي بخصوص در زمينه نرم افزارشده است. تا حدي كه مشاهده ميشود اكثر كتب درسي در عمل كاربردي ندارند. كامپايلرها نيز از اين قاعده مستثني نيست. براي نمونه مرجع 795 صفحه اي اصلي كامپايلرها [5] كه در سال 1986انتشار يافته در

عمل مطالبي از قبيل بهبود از خطا[1] و طريق عملي پياده سازي يك كامپايلر را كتمان ميكند. در اين كتاب نيز هيچگونه مطلبي در مورد يكي از روشهاي رايج براي تجزيه گرامري جملات در برنامه هاي مورد كامپايل ، بنام كاهينه بازگشتي يا در اصطلاح خارجي Recursive Descent به ميان نيامده است.

مسئله بهبود از خطا در مراجع [6] ، [8] و حاصل تحقيقي كه بر روي چند مقاله در اين زمينه انجام شد بطور عملي مشخص شده و در فصول 4 و 5 از اين كتاب گنجانده شده است. روش كاهينه بازگشتي ومسإله بهبود از خطا در مرجع [9] مطرح شده است. در اين مرجع هچگونه مطلبي در مورد مسئله بهبود از خطا و توليد كد مياني ارائه نشده است.

روش كاهينه بازگشتي براي پياده سازي كامپايلر ساده اي بنام دلتا در مرجع [10] بصورت جامع و كامل تري مطرح شده است. كمبود اين كامپايلر، مسئله بهينه سازي يا در اصطلاح Optimiztion در كد ميباشد. در اين كتاب گرامر زبان دلتا تكميل تر شده ، مسإله بهينه سازي كد نيز در اين كامپايلر در نظر گرفته شده است. كد كامپايلر دلتا بطور كامل در انتها ارائه شده است.

اصولا"، بهينه سازي [13, 15, 17, 18] در كامپايلرها در جهت افزايش سرعت اجرايي برنامه ها و اندازه اشغالي از حافظه است. در مرجع [13] روشي جديد و عملي براي پياده سازي الگوريتمهاي بهينه سازي در كد برنامه ها ارائه شده است. در فصل 8 از اين كتاب روشي جديدتر و سريع براي پياده سازي الگوريتم هاي بهينه سازي مطرح شده است. اين روش بطور عملي در كامپايلر دلتا پياده سازي شده است.

تعيين فرم كلي لغات در زبانهاي برنامه سازي بطور خلاصه و بدون ابهام توسط نوعي خاص از گراف بنام ماشين خودكار يا Finite Automata و نوعي خاص از عبارات بنام عبارات با قاعده بيان ميشود. ايجاد ماشينهاي خودكار براي بيان ساختارن لغوي زبانهاي برنامه سازي در مرجع [16] به طرز مطلوبي بيان شده است. اين روش در فصل دوم از اين كتاب تحت عنوان تحليل گر لغوي يا Lexical Analyxer مطرح شده است.

[1] با مشاهده يك جمله كه داراي اشتباه از لحاظ دستورالعمل زبان برنامه نويسي است ، كامپايلر نيز در تشخيص جملات بعدي ممكن است موجب اشتباه شود. لذا، بهبود از خطاء يا Error Recovery براي پياده سازي كامپايلرها در عمل بسيار مهم ميباشد.

مراجع

1- “SuperCompilers for Parallel and vector Computers”, Hans Zima, Acm Press, 1990, 376 pages.
2- “Parallel Programming and Compilers”, Constatine D. Polychronopoulos, Kluwer Academic Publishers, 1988, 230 Pages
3- “Symbolic Analysis For Parallelizing Compilers”, Mohamad R. Hagigat, KAP Publisher, 1885, 196 Pages.
4- “Interprocedural Def-Use Associations For C systems With Single Level Pointers”, Hermit D. Pande, IEEE Transaction On Soft. Eng., vol20, No.5, May 1994.
5- The Theory and Practice of Compiler Writing”, Paul G. Sorenson, Mc GrawHill, 1985
6- “Compilers Principles , Techniques and Tools”, Alfred V. Aho”, Addison- Wesley, 1986.
7- “Design of Compiler Techniques of Programming Language Translation”, Karen A. Lemone, CRC Press, 1992.
8- “Generation of Interactive Parsers With Error Handling”, E. Steegmans, IEEE Transaction on Software Engineering, vol. 18, no. 5, 1992.
9- “Brinch Hanson on Pascal Compilers”, Brinch Hanson, Printice-Hall, 1985.
10- “Programming Language Processors”, C.A.R Hoare, Printice-Hall, 1993.
11- “Compiler Design in C”, Allen L. Hollub, 1990.
12-“Object Oriented Compiler Construction”, Jim Holems, Printice Hall, 1995.
13- “ The Art of Compiler Design”, Thomas Pittman, Prentice-Hall, 1992.
14-“Introduction to Compilers”, Thomas W,parsons , Computer Science Press, 1992.
15-“Optimizing Schemes For Structured Programming Language Processors” , Tatsou Tsuji, Ellis Horwood, 1994.
16- “Compiler Construction, Theory and Practice”, W. A. Barrel, Science Research Associates INC, 1986.
17- “Interprocedural Def-Use Associations for C Systems with Single level Priorities. H. D. Pande, IEEE Transaction \s on Soft. Eng. , vol 20, may 1994.
18-“ Analysing the optimiztion techniques Compilers use to Transform your C Code”, Microsoft Systems Journal, March 1991.