Design and Development of Compiler  
for C- Language

**Phase IV**

**Design and Implementation**

**of Code Generator**

**Project Result**

과목명: CSE4120 기초 컴파일러 구성

담당 교수: 서강대학교 컴퓨터공학과 정성원

개발자: 1조 20141500 권태국, 20141570 이태승, 20141589 최광희(팀장)

제출일: 2017. 6. 28.

1. **개발 목표**

전체 C- Compiler 설계 프로젝트 중 Phase IV의 개발 목표는 Code Generation of C- Language로, C- Language로 쓰인 소스 코드를 MIPS Processor를 위한 Assembly Language를 Generate 할 수 있도록 한다. 구체적으로, MIPS Simulator인 SPIM에서 load 후 run할 수 있는 Assembly code를 생성하는 것을 목표로 한다.

1. **개발 범위 및 내용**

**1. 개발 범위**

Phase IV는 Phase III에서 구현한 C- Syntax tree를 이용하여, Code generator을 구현하는데, 결과적으로 완결된 Assembly code을 출력할 수 있어야 한다. 이전 프로젝트에서 Symbol table에서 주는 정보의 경우 모두 Syntax tree에 간접적으로 SymbolInfo Struct를 통해 들어가 있다. 구체적으로, Assembly code를 generate해주기 위하여 MIPS Machine을 이해해야 하는데, 해당 machine의 메모리 구조나 System call, register 사용 규약 등을 이해하고, 해당 machine에 알맞게 Assembly code를 출력해주어야 한다.

**2.개발 내용**

C- 언어는 C 언어의 Subset으로써, 단순한 문법 구조를 갖고 있다. 이번 Phase에서는 먼저, 해당 언어의 분석을 통하여 BNF Grammar로 만들어진 Syntax tree의 각 node와 1:1로 대응되는 Assembly code를 작성할 수 있어야 한다. 구체적인 출력은 파일으로, 만약 test.c 파일이 들어왔다면 test.s 파일이 출력되어야 하며, 해당 파일은 MIPS Assembly code여야 한다. 또한, System call을 사용하여 input과 output 역시 제어할 수 있어야 한다.

전체 Code generator의 개발은 크게 2단계로 수행된다. 첫 번째 단계는 각 Syntax tree의 node가 어떤 Assembly code로 generate될지를 먼저 assembly code로 hard-coding한 뒤에, 두 번째 단계에서 해당 code 를 실제로 코드로 적용시켜 자동화하는 단계이다. 그 과정에서 필요한 C- 언어로 이뤄진 예제 코드 역시 작성한다. 해당 코드는 에러가 없어야 하며, 존재하는 여러 메모리 location들과 모든 C- 언어의 syntax를 광범위하게 사용해야 한다.

1. **추진 일정 및 개발 방법**

**1. 추진 일정**

6/4 ~ 6/13 : 팀원간 역할 분배 및 이전 Project의 개발 내역을 합성 및 Refactor

6/13 ~ 6/21 : 시험 기간

6/22 ~ 6/24 : Code Generation의 이론 문서화 및 draft 작성

6/24 ~ 6/27 : 테스트 및 코드 품질 향상.

6/28 : 보고서 작성 및 제출.

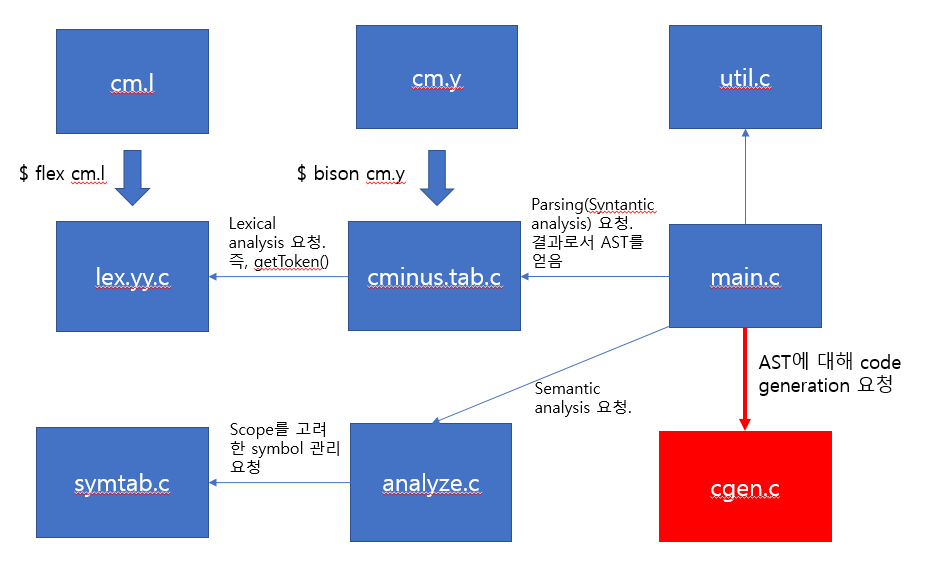
|  |
| --- |
| commit 0699e48fbf4d42663e5b2d10b415f8fc8320a30a  Author: juice500ml <juice500ml@gmail.com>  Date: Sun Jun 4 17:20:30 2017 +0900  DONT\_OCCUR\_PRINT only in DEBUG mode  commit 1c81de368610be2c96a0fdc7fd947c9f0c00a560  Author: Taeguk Kwon <xornrbboy@gmail.com>  Date: Sun Jun 4 01:43:58 2017 -0700  Refactor (#19)  commit c198e540c26ad4aef08926eea1633da96b6483bf  Author: juice500ml <juice500ml@gmail.com>  Date: Sun Jun 4 17:43:58 2017 +0900  Refactor memory allocation errors (#19)  commit 04d9719141d71bf3d551d29aefb60bc23a6777f6  Merge: 1c81de3 c198e54  Author: Taeguk Kwon <xornrbboy@gmail.com>  Date: Sun Jun 4 01:47:10 2017 -0700  Resolve conflict.  commit 4738f06e6237d6dc7c5dfff48a402f961ac5c7f7  Author: juice500ml <juice500ml@gmail.com>  Date: Sun Jun 4 17:47:22 2017 +0900  Caught bug in log\_debugger  commit 881809820d13a9c2de260e2f97762ab8caa9036f  Merge: 4738f06 04d9719  Author: juice500ml <juice500ml@gmail.com>  Date: Sun Jun 4 17:47:38 2017 +0900  Merge branch 'master' of https://github.com/juice500ml/compiler-sogang  commit 9ce23806918efe172da7fbb2a289e4c987618844  Author: juice500ml <juice500ml@gmail.com>  Date: Sun Jun 4 17:49:26 2017 +0900  Caught bug in log\_debugger  commit 38e7135880064aa902d99bc3a4589e9961177ada  Author: Taeguk Kwon <xornrbboy@gmail.com>  Date: Sun Jun 4 01:59:51 2017 -0700  Refactor (#19)  commit 868cabc2591fd4b9c4a8e79d094ba260445cb66f  Merge: 38e7135 9ce2380  Author: Taeguk Kwon <xornrbboy@gmail.com>  Date: Sun Jun 4 02:00:04 2017 -0700  Merge branch 'master' of https://github.com/juice500ml/compiler-sogang  commit 856d23f4e8b52971191a25a3432d9d83c4262b48  Author: Taeseung Lee <taeseunglee159@gmail.com>  Date: Sun Jun 4 02:08:57 2017 -0700  refactor printSymtab  commit 5602391a36da79d2f60c499f358d0b563602511f  Author: Taeguk Kwon <xornrbboy@gmail.com>  Date: Sun Jun 4 02:09:07 2017 -0700  Refactor  commit 5d1d5e552386f7dc47076db6782adba52b923d76  Merge: 856d23f 5602391  Author: Taeseung Lee <taeseunglee159@gmail.com>  Date: Sun Jun 4 02:09:29 2017 -0700  Merge branch 'master' of https://github.com/juice500ml/compiler-sogang  commit 448aa580bfba797d03ba7049221b9000edc81488  Author: Taeseung Lee <taeseunglee159@gmail.com>  Date: Sun Jun 4 02:14:26 2017 -0700  span Name space in printSymtab  commit f02caf7f5dc097113cade9d3bf0e4742f6b95c68  Author: juice500ml <juice500ml@gmail.com>  Date: Sun Jun 4 18:31:06 2017 +0900  Refactor (#19)  commit 99cd07d82c6fc1485e63ed30f18cedaa700ea796  Merge: f02caf7 448aa58  Author: juice500ml <juice500ml@gmail.com>  Date: Sun Jun 4 18:31:30 2017 +0900  Merge branch 'master' of https://github.com/juice500ml/compiler-sogang  commit d8ef0780ebe1a5f25609a4529a01cd4f72164bd4  Author: juice500ml <juice500ml@gmail.com>  Date: Sun Jun 4 21:57:59 2017 +0900  Fix #11  commit ec48995f50996530d1a98426286c0a9d6301f7d7  Author: juice500ml <juice500ml@gmail.com>  Date: Sun Jun 4 21:59:44 2017 +0900  Fix #15  commit 4819ba8c264ca39b6e5f63e33b1108ce3b14624a  Author: Taeseung Lee <taeseunglee159@gmail.com>  Date: Sun Jun 4 06:03:58 2017 -0700  Remove useless TODO  commit f63cec06b6205c45eb2065f17055cf585cf99817  Author: Kwanghee Choi <juice500ml@gmail.com>  Date: Sun Jun 4 06:40:36 2017 -0700  Copy TINY code  commit 68e152710cffbd58ea7203cb6f0067cc0b71d656  Merge: f63cec0 4819ba8  Author: Kwanghee Choi <juice500ml@gmail.com>  Date: Sun Jun 4 06:40:54 2017 -0700  Merge branch 'master' of https://github.com/juice500ml/compiler-sogang  commit 87e29999192d6629e245fba2660f234ca100301e  Author: Taeseung Lee <taeseunglee159@gmail.com>  Date: Mon Jun 5 01:02:45 2017 -0700  Set variables and a define for stating project4 Code Generation  commit fb900f46a1193bcd605dc3b7c2680d791b3ff3fa  Author: Taeseung Lee <taeseunglee159@gmail.com>  Date: Mon Jun 5 07:21:36 2017 -0700  Change len info into arrLen and paramLen  commit ad5dc0a481f0e96ecc4f084a7c40d583a6ce6aa2  Author: Kwanghee Choi <juice500ml@gmail.com>  Date: Wed Jun 21 07:20:55 2017 -0700  Prototype for global decls  commit bf3d801f649dff4a5578f47ecb56dbdb984d2835  Merge: ad5dc0a fb900f4  Author: Kwanghee Choi <juice500ml@gmail.com>  Date: Wed Jun 21 07:22:36 2017 -0700  Merge conflict of arrLen resolved  commit 05da5bff34a3bd6ffe598bfd5647f09dc65a7d74  Author: Kwanghee Choi <juice500ml@gmail.com>  Date: Thu Jun 22 01:42:27 2017 -0700  Change SymbolInfo structure. Use globalFlag and stop using label for functions  commit 4c5693fafa12f2343843730aaa8cf7a303691033  Author: Kwanghee Choi <juice500ml@gmail.com>  Date: Thu Jun 22 02:33:18 2017 -0700  Output asm file name is now correct  commit 714c393bfed24c9218f24347d1bb0caaff804e5d  Author: Kwanghee Choi <juice500ml@gmail.com>  Date: Thu Jun 22 07:43:11 2017 -0700  Draft for project 4  commit 0ccf615ce2e654910946197f6a9a38eccc679d12  Author: Kwanghee Choi <juice500ml@gmail.com>  Date: Thu Jun 22 07:52:37 2017 -0700  Add return statement calculation  commit 1ae218502f7adf61135021ea1d8feaab9762d69f  Author: Kwanghee Choi <juice500ml@gmail.com>  Date: Sat Jun 24 04:17:09 2017 -0700  issue #21 fix  commit f070968f9440857decfdfe5cab14671daf5579a7  Author: Kwanghee Choi <juice500ml@gmail.com>  Date: Sat Jun 24 08:22:11 2017 -0700  Implement function call stack, but not yet tested  commit a07051e283990d13d8f6ab874b82909c92f55525  Author: Kwanghee Choi <juice500ml@gmail.com>  Date: Sat Jun 24 08:36:09 2017 -0700  caught typo bug  commit e59a5ae4a6e8c73b2e932debebe0e7d7f13052db  Author: Kwanghee Choi <juice500ml@gmail.com>  Date: Sat Jun 24 08:42:28 2017 -0700  Caught typo bug, changed comment value (#23 No need for fixing CoumpoundStatementK block)  commit d6b948d528d6f3e6d4e87bf6c9b07735739c7207  Author: Kwanghee Choi <juice500ml@gmail.com>  Date: Sat Jun 24 08:47:30 2017 -0700  No need to check 0 of stack size (#23)  commit 030a4c759c47354114544a2df53e4c1e03dde578  Author: Taeseung Lee <taeseunglee159@gmail.com>  Date: Sun Jun 25 06:11:40 2017 -0700  Add symbols for input and output function and testcase #24  But this doesn't catch the exceptional case  commit 75274df99a99227735f5d86698eb3dc9c96c791f  Author: Taeseung Lee <taeseunglee159@gmail.com>  Date: Sun Jun 25 23:54:36 2017 -0700  Add testcases for project4  commit 2bef244097f7ba5e6314928efa50b5ee72409aab  Author: Taeseung Lee <taeseunglee159@gmail.com>  Date: Mon Jun 26 00:07:43 2017 -0700  Fix some bugs and Add .data code  commit fad4f8379a6e47f165fe25ee058e6b0ca315b54f  Author: Taeseung Lee <taeseunglee159@gmail.com>  Date: Mon Jun 26 00:10:34 2017 -0700  Add a generater for input() and output() to cgen.c  commit b1a23eb4398e26ada7ff78c13579bc1f0cf7edcd  Author: Taeseung Lee <taeseunglee159@gmail.com>  Date: Mon Jun 26 00:23:15 2017 -0700  Remove \_\_ prefix files  commit e58c7dbc9d1d25ad5f3a4ae5f7a9b583d92d57db  Author: Taeseung Lee <taeseunglee159@gmail.com>  Date: Mon Jun 26 00:37:32 2017 -0700  Add two testcases for project4  commit 8016fae9d30626dae9277e7b914c955577a0a355  Author: Taeguk Kwon <xornrbboy@gmail.com>  Date: Mon Jun 26 02:40:39 2017 -0700  Fix #25.  commit 010ecc1244eb79683f8bfe67b39f8257fb066d47  Merge: 8016fae e58c7db  Author: Taeguk Kwon <xornrbboy@gmail.com>  Date: Mon Jun 26 02:40:58 2017 -0700  Merge branch 'master' of https://github.com/juice500ml/compiler-sogang  commit a5f5de5334a08d26d2b63695322589d46f399fe6  Author: Taeguk Kwon <xornrbboy@gmail.com>  Date: Mon Jun 26 02:52:17 2017 -0700  Fix a bug in ArrayK. And cleanup stackframe of function in FunctionDeclarationK (#23).  commit 0088fb0788de4a0b4f6cf274e3bd0ea7e79a22c3  Author: Taeseung Lee <taeseunglee159@gmail.com>  Date: Mon Jun 26 03:02:36 2017 -0700  Fix #22 and small bugs  commit 0cd487a0f7262ce163099029d04ef4be5a3ca3d9  Author: Taeguk Kwon <xornrbboy@gmail.com>  Date: Mon Jun 26 06:21:27 2017 -0700  Handle several return statements (#23).  commit 6f2eefa4b086b8057f94b81e633d8e6b19d5a61f  Author: Taeguk Kwon <xornrbboy@gmail.com>  Date: Mon Jun 26 06:49:39 2017 -0700  Fix a bug in ArrayK.  commit a8440a8e33e7aacc93581a14fd2ba677953c9f5a  Author: Taeguk Kwon <xornrbboy@gmail.com>  Date: Mon Jun 26 07:41:04 2017 -0700  Fix a bug about parameter passing. And fix a mistake about intInfo and arrINFO.  commit af2c7bb67d7ea13b4097cfeb0cc63b527687b369  Author: Taeguk Kwon <xornrbboy@gmail.com>  Date: Mon Jun 26 08:04:13 2017 -0700  Fix a bug about accessing an element of array which is parameter.  commit 0ee72eceaa1c64b4ca7084d56a6f80bfc6bec305  Author: Taeseung Lee <taeseunglee159@gmail.com>  Date: Mon Jun 26 08:17:15 2017 -0700  Add Boss testcase  commit c2bf6604ab1ffe19bd36ac1fd5f879d910a17888  Author: Taeguk Kwon <xornrbboy@gmail.com>  Date: Mon Jun 26 08:48:50 2017 -0700  Fix a mistake in boss.c  commit abf021ecc2d58dc3c899826a967d3c51cb3b9013  Author: Taeguk Kwon <xornrbboy@gmail.com>  Date: Mon Jun 26 08:50:04 2017 -0700  Fix bugs for testcase boss.c.  commit 1ebfc6ee4b868e6078ec2c25647afe633f855855  Author: Taeseung Lee <taeseunglee159@gmail.com>  Date: Mon Jun 26 09:00:31 2017 -0700  Update boss testcase and Complete tests  commit 5d7a665afc9b9964b3abc62a67d1eb604906d4ce  Author: Kwanghee Choi <juice500ml@gmail.com>  Date: Mon Jun 26 23:10:21 2017 -0700  Bug on ConstantK caught  commit 98c13369217f35170c0d9141a43b8cd8aa890f9c  Author: Kwanghee Choi <juice500ml@gmail.com>  Date: Tue Jun 27 01:17:09 2017 -0700  testcase 6  commit 5db658404c1c1a6789e656044c6ef93bd7767a81  Author: Kwanghee Choi <juice500ml@gmail.com>  Date: Tue Jun 27 01:17:57 2017 -0700  typo for example 6  commit 05f9ed5eb046045a3965270c09093475cb5f4d88  Author: Taeguk Kwon <xornrbboy@gmail.com>  Date: Tue Jun 27 03:35:14 2017 -0700  Fix a bug about code generation. |

**2. 개발 방법**

개발 환경은 cspro9와 cspro10을 사용하여, Ubuntu 16.04.2 LTS 기준으로 작업한다. vi editor을 이용하여 편집을 하며, Indentation은 GNU Standard를 따른다. 주 개발 언어는 C이나, 이번 Phase의 경우 Syntactic Specification 소스 역시 작성한다. 컴파일은 gcc 5.4.0 을 통해 하며, Makefile을 이용하여 빌드 및 Testing의 자동화를 한다. git을 이용하여 Version Control을 하며, Github에서 remote branch를 관리하여, 매 Commit마다 각 case들에 대해 검증한다.

1. **연구 결과**

**1. 합성 내용**



전체 소프트웨어 분석도

**2. 분석 내용**

이번 프로젝트에서는 크게 두 부분으로 나누어 개발했다. 먼저, 각 Syntax tree의 node가 어떤 Assembly code로 generate될지를 먼저 assembly code로 hard-coding을 하고, 해당 code 를 실제로 코드로 적용시켜 자동화했다. 여기에서는 두 번째 부분을 실제로 어떻게 구현했는지를 중심으로 살펴보겠다.

개념적으로 cgen.c를 어떻게 구현했는지 살펴본다면 다음과 같다.

codeGen 함수를 부르면, 해당 Syntax Tree가 codeStream에 write되게 된다. 먼저, codeGen에서는 Scope가 전역인 경우를 처리하고, local의 경우 localCodeGen 함수를 통해 처리한다. 이 두 함수에서 필요한 여러 util함수인 globalMemAlloc, labelAlloc 및 util 변수 L\_cleanup 역시 개발했고, 이들 모두 cgen.c에 위치한다.

* MIPS Memory Management  
  Global memory의 경우 0x10000000에서 증가하는 방향으로 쌓인다.  
  Local memory의 경우 $(sp)에서 감소하는 방향으로 쌓인다.
* Run entry point  
  spim에서 run을 실행할 경우, main 함수가 implicit하게 불리므로, entry point는 main label이 된다.
* 사용할 수 있는 Instruction들은 제시된 SPIM Manual에 정리되어 있다.

**3. 제작 내용**

이번 프로젝트에서 크게 잡은 함수들을 중심으로 제작 내용을 소개하도록 하겠다.

* 기존의 프로젝트에서 수정한 내용

기존 프로젝트에서 몇몇 버그를 제외하고는 거의 동일하나, int input(void)와 void output(int) 두 system call이 implicit하게 추가되었으므로, 해당 두 함수를 syntax tree에 넣어주는 registerIO() 함수가 analyze.c에 추가되어 있다. 또한, cgen.c / cgen.h 두 파일로만 이번 프로젝트의 전체 code generation을 진행하므로, 이에 대해서는 밑에서 설명하도록 하겠다.

* void codeGen(TreeNode \*, FILE\*)  
  codeGen()의 경우, 전역인 변수들, 측 Variable, Array, Function 총 세 개의 node에 대해서 code generation을 진행해야 한다. 이들 중 Variable과 Array는 globalMemAlloc 함수를 통하여 memory를 allocate해주어 memloc 변수 안에 넣어놓고, Function의 경우 Stack을 사용하여 Parameter passing 과정을 implicit하게 진행이 되었다고 가정을 하고 사용하는데, 이전 register의 값을 저장해놓기 위해 stack을 사용하므로, stack의 사용과 cleanup 코드가 Function의 code generation의 맨 위와 맨 아래에 위치하게 된다. 여기서, L\_cleanup이라는 변수를 사용하게 되는데, 해당 label이 곧 stack cleanup의 시작을 알리므로, 이후 return statement에서 유용하게 쓰이게 된다.
* static int localCodeGen(TreeNode \*, FILE \*, int currStack, int travSibling)  
  해당 함수가 처음 call되는 시점은 codeGen의 Compound statement로, 해당 function의 compound statement를 작성하는 함수라고 봐도 무방하다. 이는 C minus 언어가 함수 바깥에 존재하는 statement들이 없기 때문이다. 이 함수는 재귀적으로 code를 generate하는데, 이전 nodeKind에서 정의되던 모든 Kind들에 대해 1:1로 대응되는 코드를 generate해준다. 기본적인 구조는, 필요한 경우 자신의 child를 먼저 generate해주기 위하여 이 함수를 재귀적으로 call을 해준 뒤에, 규약에 의하여 Register $v0에 넘겨온 result를 해당 node에서 적당한 값으로 manipulate해주는 것이다.
* memloc에 저장된 값  
  memloc에는 type별로 저장되는 값이 다른데, global의 경우 실제 메모리가 들어가게 되므로 globalFlag = TRUE로 세팅이 된다. local의 경우 Register $fp를 기준으로 memloc($fp)가 이 변수의 주소가 된다. 이는, $sp는 코드가 진행됨에 따라 계속 변하기 때문이다. 이는 VariableK case에서 명확하게 드러난다.
* Parameter passing  
  Parameter passing의 경우 CallK case에서 볼 수 있듯, input / output을 제외한 함수들의 경우, Caller가 각 parameter에 대해 stack에 차례대로 push를 했는데, callee는 그러므로 0($fp), 4($fp) … 와 같이 해당 parameter을 접근할 수 있다.
* CompoundStatementK  
  이 Kind의 경우 중간에 변수가 선언되면 stack이 바뀌기 때문에, 현재 node의 자식들인 variableDeclarationK에서 stack growth를 해주었고, local stack cleanup은 해당 함수의 return 값을 통하여 cleanup을 진행해준다.

**4. 시험 내용**

여러 사항에 대해서 여러 테스트 케이스를 만들어 시험했다.

|  |
| --- |
| void main(void){  int a;  int b;  int c;  int d;  a = 1;  b = 2;  c = 3;  d = 4;  output(a+b);  output(a-b);  output(d/b);  output(d\*b);  a = a + b + ((c + d) + 5 + 6) + 7 + 8 \* 9;  output(a);  b = c \* d - b;  output(b);  c = d / (c - 1);  output(c);  d = a / b - c \* c;  output(d);  } |
| Filename: exam1.c |
| (spim) load "exam1.s"  (spim) run  Output : 3  Output : -1  Output : 2  Output : 8  Output : 100  Output : 10  Output : 2  Output : 6 |

|  |
| --- |
| /\* int save[1001]; \*/  int fibonacci(int a)  {  int ret;  if(a <= 0) ret = 0;  else {  if(a <= 2) ret = 1;  else {  ret = fibonacci(a-1) + fibonacci(a-2);  }  }    return ret;  }  void main(void)  {  int ret;  ret=input();  output( fibonacci(ret) );  } |
| Filename: exam2.c |
| (spim) load "exam2.s"  (spim) run  Input : 6  Output : 8  (spim) run  Input : 7  Output : 13  (spim) run  Input : 8  Output : 21 |

|  |
| --- |
| int save[1001];  int fibonacci(int a) {  if(a <= 0)  return 0;  else if(a <= 2)  return 1;  {  int k;  k=a;  {  int y[10];  y[5]=0;  if(save[k]!=y[5]) return save[k];  }  {  int h;  h=fibonacci(k-1);  {  int g;  g=fibonacci(k-2);  save[a]=h+g;  {  int aaa[400];  aaa[333]=save[a];  return aaa[333];  }  }  }  }  }  void main(void) {  int ret;  ret=input();  output( fibonacci(ret) );  /\* ret must has 610 : 0x0262\*/  } |
| Filename: exam3.c |
| (spim) load "exam3.s"  (spim) run  Input : 8  Output : 21  (spim) run  Input : 9  Output : 34  (spim) run  Input : 10  Output : 55 |

|  |
| --- |
| /\* A program to perform Euclid's  Algorithm to compute gcd. \*/  int gcd (int u, int v)  {  if (v == 0)  return u;  else  return gcd(v, u-u/v\*v);  /\* u-u/v\*v == u mod v \*/  }  void main(void)  {  int x;  int y;  x = input();  y = input();  output(gcd(x, y));  } |
| Filename: exam4.c |
| (spim) load "exam4.s"  (spim) run  Input : 3  Input : 6  Output : 3  (spim) run  Input : 10  Input : 22  Output : 2  (spim) run  Input : 84  Input : 48  Output : 12 |

|  |
| --- |
| /\* A program to perform selection sort on a 10 element array. \*/  int x[10];  int minloc (int a[], int low, int high)  {  int i;  int x;  int k;  k = low;  x = a[low];  i = low+1;  while (i < high)  {  if (a[i] < x)  {  x = a[i];  k = i;  }  i = i+1;  }  return k;  }  void sort (int a[], int low, int high)  {  int i; int k;  i = low;  while (i < high-1)  {  int t;  k = minloc(a, i, high);  t = a[k];  a[k] = a[i];  a[i] = t;  i = i+1;  }  }  void main(void)  {  int i;  i = 0;  while (i < 10)  {  x[i] = input();  i = i+1;  }  sort(x, 0, 10);  i = 0;  while (i < 10)  {  output(x[i]);  i = i+1;  }  } |
| Filename: exam5.c |
| (spim) load "exam5.s"  (spim) run  Input : 3  Input : 6  Input : 7  Input : 5  Input : 63  Input : 12  Input : 534  Input : 7  Input : 3  Input : 7  Output : 3  Output : 3  Output : 5  Output : 6  Output : 7  Output : 7  Output : 7  Output : 12  Output : 63  Output : 534 |

|  |
| --- |
| int N;  int X[21];  int Y[21];  int ans;  void dfs(int x, int y, int p, int m, int n) {  if (p == N / 2) {  int i;  i = n;  while(i<N) {  x = x - X[i];  y = y - Y[i];  i = i + 1;  }  if (ans > x\*x + y\*y)  ans = x\*x + y\*y;  return;  }  if (m == N / 2) {  int i;  i = n;  while(i<N) {  x = x + X[i];  y = y + Y[i];  i = i + 1;  }  if (x\*x + y\*y)  ans = x\*x + y\*y;  return;  }  dfs(x + X[n], y + Y[n], p + 1, m, n + 1);  dfs(x - X[n], y - Y[n], p, m + 1, n + 1);  }  void main(void) {  int tc;  int T;  T = input();  while(T) {  int i;  ans = 987651345;  N = input();  i = 0;  while(i<N) {  X[i] = input();  Y[i] = input();  i = i + 1;  }  dfs(0, 0, 0, 0, 0);  output(ans);  T = T - 1;  }  return;  } |
| Filename: exam6.c |
| (spim) load "exam6.s"  (spim) run  Input : 2  Input : 3  Input : 1  Input : 1  Input : 2  Input : 2  Input : 3  Input : 3  Output : 32  Input : 10  Input : 6  Input : 8  Input : 3  Input : 6  Input : 1  Input : 11  Input : 4  Input : 3  Input : 7  Input : 9  Input : 3  Input : 1  Input : 6  Input : 9  Input : 9  Input : 8  Input : 7  Input : 2  Input : 21  Input : 4  Output : 794 |

|  |
| --- |
| int globalA;  int globalArr[20];  int helloWorld(int a)  {  int b;  a = a+1;  b = 5;  {  int a;  a = b;  }  return a;  }  int sumTwo(int a, int b)  {  return a + b;  }  int sumThree(int a, int b, int c)  {  return a + b + c;  }  int sumFour (int a, int b, int c, int d)  {  return a + b + c +d;  }  int polynomial(int n, int arr[], int x)  {  int res;  res = 0;  while (n > 0)  {  n = n-1;  res = res \* x + arr[n];  }  return res;  }  void helloGlobal(int globalA, int b)  {  globalA = b; /\* fake globaA \*/  globalArr[0] = b;  }  void checkGlobalA(void)  {  int i;  i = globalA;  while (i)  {  output(globalA);  i = i-1;  }  }  void main(void)  {  int a;  int b;  int c;  int arr[3];  int testArr[50];  testArr[(a = input()) \* 2]  = c  = arr[0]  = testArr[(b = input()) \* 3]  = c  = arr[0]  = arr[0]  = testArr[0]  = input();  output(a);  output(b);  output(c);  output(arr[0]);  output(testArr[0]);  output(testArr[a\*2]);  output(testArr[b\*3]);  arr[0] = a = input();  arr[1] = input();  arr[2] = input();  /\* result: a+2 \*/  output(helloWorld(helloWorld(a)));  /\* result: sum~ \*/  output(sumTwo(a, arr[0]));  output(sumTwo(arr[1], a));  output(sumTwo(  sumTwo(3, 5),  sumTwo(5, 7)  ));  output(sumThree(arr[0], arr[1], arr[2]));  output(sumFour(a, arr[0], arr[1], arr[2]));  globalA = 0;  {  int a;  int b;  int globalA;  globalA = 5;  }  /\* helloGlobal \*/  output(globalA);  globalA = 0;  helloGlobal(globalA, 5);  output(globalA);  output(globalArr[0]);  {  int res;  int arr[5];  output (888888); /\* polynomial mark \*/  arr[0] = 1;  arr[1] = 2;  arr[2] = 3;  arr[3] = 4;  arr[4] = 5;  res = polynomial(5, arr, arr[0]);  output(res);  }  /\* if statement \*/  if (globalA < 2) /\* always true \*/  {  output(5);  }  else  output(99999999);  if (5 > 3) /\* always true \*/  {  output(55);  }  if (5 == 5)  {  output(555);  }  if (a > a-1)  {  int a;  a = 33;  output(a); /\* 33 \*/  if (a != a-4)  {  int c;  c = a;  output(c); /\* 33 \*/  while(0);  }  }  globalA = 3;  while (globalA)  {  checkGlobalA();  globalA = globalA-1;  }  } |
| Filename: boss.c |
| (spim) load "boss.s"  (spim) run  Input : 1  Input : 2  Input : 3  Output : 3  Output : 2  Output : 1  Output : 1  Output : 1  Output : 1  Output : 1  Input : 1  Input : 2  Input : 3  Output : 3  Output : 2  Output : 3  Output : 20  Output : 6  Output : 7  Output : 0  Output : 0  Output : 5  Output : 888888  Output : 15  Output : 5  Output : 55  Output : 555  Output : 33  Output : 33  Output : 3  Output : 3  Output : 3  Output : 2  Output : 2  Output : 1 |

**5. 평가 내용**

개발시에 프로그램에 존재하는 버그를 제거하기 편하게, 여러 가지 장치를 했다. 첫번째로, makefile에 all 이외에 debug, warn 이라는 target을 추가적으로 구현했다. Make debug의 경우에는 -g 옵션(gdb)와 -D DEBUG(Define DEBUG)를 했다. 이를 통해, make debug만 하면 바로 gdb를 사용할 수 있는 실행 파일이 만들어진다. 또한, -D DEBUG를 이용해서, log\_debugger.h에 있는 LOG\_DEBUG macro function이 정보들을 출력해준다. 정보의 종류에는 LOG\_DEBUG가 출력하고자 하는 메세지, 그리고 메세지에 출력 변수로 들어갈 변수들이 있으며, LOG\_DEBUG가 있는 파일, line number, 함수를 출력하여, 어디서 문제가 발생했는지 쉽게 알 수 있게 한다. -D DEBUG 옵션이 없는 경우에는 LOG\_DEBUG는 아무런 출력을 하지 않는다. 이를 통해, 개발을 편하게 하였다. 그리고 절대로 발생하지 말아야하는 경우(예를 들면, malloc의 할당 실패 또는 switch case에서 처리하지 않아야하는 default case, 잘못된 값이 들어옴을 의미합니다.)에 대해서 DONT\_OCCUR\_PRINT macro를 넣어줬다. 이 매크로는 절대 발생하지 않아야하는 부분이 발생했을 때, -D DEBUG옵션과 같이, 파일, 라인, 함수과 같이 에러 메세지를 출력해준다. 이를 통해, 좀 더 빠르게 버그 위치를 확인할 수 있고 고치는 데 유용하였다. Make warn의 경우에 런타임 에러가 발생할 수 있는 가능성을 줄이기 위해, gcc에 warning 옵션을 추가해준다. 여기에 많은 warning 옵션들이 들어가는데 이를 입력해주는 수고를 덜어주며, 리팩토링을 쉽게 해준다.

또한, 개발시에 Git을 사용하여 Commit을 잘게 쪼개 각 변경마다 어떤 사항이 변경되었는가를 Tracking 하도록 하여, 신뢰성 있는 Code 작성 및 Continuous Integration 뿐만 아니라 과제를 제출한 뒤에 지속적으로 타인들과 함께 Open Source Contribution을 할 수 있도록 했다. 또한 문제가 생겼을 때 Revert나 Cherry-pick 등 여러 Version Control 기능을 사용할 수 있어, 개발을 훨씬 안정적으로 할 수 있었다. 해당 코드를 Github에 올림으로써, Local Repository에서만 관리하는 것이 아닌 Remote Repository를 두어, 한 쪽 Code가 날아가더라도 전혀 지장이 없도록 했다.

1. **기타**
2. **연구 조원 기여도**

20141589 최광희 (34%)

20141570 이태승 (33%)

20141500 권태국 (33%)

1. **자체 평가**

우리 팀이 진행한 프로젝트의 경우, 설계 측면의 경우 기존에 TINY Compiler에서 code와 cgen을 쪼갠 것과는 다르게, 고민을 통해 code가 필요 없음을 깨닫고, 오직 cgen을 사용하여 개발했다. 개발 측면의 경우, make에 make debug, make warn 등 여러 버전을 준비하여 개발 중 찾아오는 여러 상황들에 대해 효과적으로 대처할 수 있도록 하였다. 또한, Makefile을 hierarchical 구조로 만들어 추후 확장될 코드들에 대해 합리적으로 빌드할 수 있도록 하였다. Test를 할 때도 주어진 예외 뿐만 아니라 여러 예외들에 대해서 모두 잡을 수 있도록 다소 많은 cminus code를 작성했고,

개발 방법론에 있어서는, 기본적으로 주어진 명세서를 기준으로 Kanban 방식을 도입하고, 크지 않은 팀원 수에 알맞게 Scrum과 Pair-programming을 섞어 사용하였다. Github의 Issue system을 통해 Root issue는 Kanban처럼 관리하고, 해당 issue에 소issue들을 생성하거나, Kanban 외의 issue 총 두 가지의 issue 종류로 쪼개, 첫 번째 종류의 issue들은 간단한 issue의 경우 발생한 당일 각자의 시간에 Scrum을 진행하고, 서로의 합의가 필요한 복잡한 issue의 경우 모여서 회의를 1~2시간가량 진행한 뒤에 개발을 진행했다. 복잡한 이슈의 경우, Pair-programming을 하여 만든 Draft를 통해 Pseudo-code를 먼저 생성을 한 뒤에 이후 해당 code에 대한 fix를 하는 방식으로 진행했다.

개발시에 Git을 사용하여 신뢰성 있는 Continuous Integration 뿐만 아니라 이후에도 지속적으로 Open Source Contribution을 할 수 있도록 했다. 문제가 생기더라도 여러 Version Control 기능을 통해 안정적인 개발을 할 수 있었다. 해당 코드는 Github를 통해 Remote Repository를 두어 한 쪽 Repo에 문제가 생기더라도 문제 없도록 했다.