컴파일러 2015 Project

Item 1: U-코드로 프로그래밍하여 제출
Item 2: mini C 코드를 입력받아 U-코드로 변환 시켜주는 translator 작성함
Item 3: 자유로운 주제를 가지고 Item 2의 결과를 컴파일러를 수정/개선해본다.

주요 일정

① (2015-11-29 일 23:59:59 까지)
Item 1에 대한 결과를 제출한다. (개인)

② (2015-12-9 수 23:59 까지)
Item 2 결과물 제출: 조교가 명시하는 대로 제출 (개인)

③ (2015-12-11 금 23:59 까지)
Item 3에 대한 설계계획서를 제출한다. (개인 또는 팀) 팀 구성을 하는 경우 역할 분담에 대한 것도 적는다.

④ (2015-12-16 수 기말고사 실시 예정)

⑤ (2015-12-21 월 23:59 까지)
Item 3 결과물 제출: 설계보고서 (개인 또는 팀) 설계 계획서가 바뀌었으면 함께 제출

Item 1: U 코드 코딩 연습

1. 다음 sum.mc를 U-코드로 (손으로) 바꾼다.

```
int result = 0;
void main(void)
{
        int i;
        int n;
        n = 19;

        for( i = 1 ; i <= n   ; i = i + 1 )
        {
                result = result +  i ;
        }

        write(result);
}
```

2. 작성한 U-코드를 U-코드 인터프리터에 입력, 결과를 확인한다.
   (제출물) 화면 캡쳐, 변환된 코드, U-코드를 분석한 보고서

Item 2 U-code 생성기

1. 과제 3을 바탕으로 U-코드를 generation 하는 프로그램을 작성한다. 아래 함수를 사용한다.

예)

```
class UcodeCodeGen {
      static void minic2ucode(String mcFile) {
          MiniCLexer lexer = new MiniCLexer( new ANTLRFileStream(mcFile));
          CommonTokenStream tokens       = new CommonTokenStream( lexer );
          MiniCParser parser             = new MiniCParser( tokens );
          ParseTree tree          = parser.program();

          ParseTreeWalker walker  = new ParseTreeWalker();
          walker.walk(new UCodeGenListener(), tree );
          }
          ...
}

    class UCodeGenListener extends MiniCBaseListener {
      ...
       ParseTreeProperty<String> newTexts = new ParseTreeProperty<String>();
       @Override public void entryProgram(MiniCParser.ProgramContext ctx) { ..
```

2. miniC 코드를 입력하고 결과로 생성된 파일을 인터프리터 위에서 수행시켜보고 확인한다.
   (제출물) 화면 캡쳐, 변환된 코드, 작성한 코드에 대한 보고서, 프로젝트 압축 파일

<코드 생성 참고> - 교재 10.4

1. 전체: void codeGen(Node *ptr)
   - U-code 인터프리터에 맞는 시작 코드 (starting code)생성 포함

2. 선언문: void processDeclaration(Node *ptr)
   void processSimpleVariable(Node *ptr, int typeSpecifier, int typeQualifier)
   void processArrayVariable(Node *ptr, int typeSpecifier, int typeQualifier)

3. 식: void processOperator(Node *ptr)

A. Assignment
B. Binary operatior
C. Uniary operator
D. Array
E. Increase, Decrease

4. 문장: void processStatement(Node *ptr)
   if, while, return, compound statement, expression statement, goto

5. 함수 processFunction()
   정의, 호출

Item 3. 자유로운 주제를 가지고 Item 2 의 결과를 컴파일러를 수정/개선해본다

예)

- mini C를 x86이나 x64, ARM 등의 어셈블리로 변환하는 컴파일러 만들기
- mini C를 GCC GIMPLE이나 LLVM IR 등 U-코드가 아닌 다른 중간 코드로 변환하는 컴파일러 만들기
- mini C나 U-코드의 제어 흐름 그래프를 만들어 화면에 보여주기
- mini C에서 함수 호출 그래프를 만들어 화면에 보여주기
- 변환된 U-코드에 최적화를 해주는 컴파일러 만들기
- 타입 검사가 추가된 mini C to U-코드 컴파일러 만들기
- 기타 수업 중 배운 기법들

제출물 (Item 3 에 대해)

① 설계 계획서

1-1 채택할 goal 및 적용 계획에 대한 토의 결과 (1page 내외)

1-2 구현 일정 계획

1-3 팀 구성 소개 및 역할 분담 내용

② 설계 보고서

2-1 채택/고안한 기법에 대한 설명 (2page 내외)

2-2 결과 요약 (1page 내외)

2-3 장단점 분석 (1page 내외)

2-4 코드

- 설계계획서, 설계보고서, 양식은 별도로 제공된다.
- 예시로 소개된 Item 2 주제들 일부에 대해서는 추후 필요에 따라 보조 설명이 있을 예정임

Item 3 팀구성

- Item 3 는 원칙적으로 2-3인으로 구성된 팀별 과제이다.
- 원하면 단독으로 할 수도 있다.
- 팀원 모두가 Item2 를 제출한 상태여야 팀구성이 가능하다.
- 팀을 구성하는 경우와 개인 단독 수행의 점수상 차이는 없으나, 단 동점일 때는 개인 단독 제출자가 더 높은 순위를 가진다.
- 팀으로 구성한 제출물에 대해서는, 역할에 대한 면밀한 조사를 하여 팀원간 역할

배분이 상식 이하로 불공평할 때는 경우에는 크게 (경우에 따라 Item 3의 배점 이하로) 감점한다. 즉, 현실적으로 두 사람 이상이 함께 시너지를 내어서 잘 할 수 있는 경우에만 전략적으로 팀으로 하여 각자의 로드를 줄이고, 그렇지 않는 경우는 절대 권하지 않는다.