Diseño de Compiladores I
Trabajo Práctico Nº 4
2018

Objetivo

Generar código Assembler para Pentium de 32 bits, a partir del código intermedio generado en el Trabajo Práctico Nº 3.
El mecanismo de generación de Código Assembler, será el que corresponda al tema particular asignado al grupo:

- Seguimiento de registros
- Variables auxiliares

El código Assembler generado debe poder ser ensamblado y ejecutado sin errores. El ensamblador a utilizar se puede descargar desde la página de la cátedra.

Conversiones implícitas:

Para los grupos que deben considerar conversiones implícitas, el código Assembler deberá efectuar dichas conversiones cuando los tipos de los datos involucrados en una expresión lo requieran, y siempre que el lenguaje lo permita.

. Los grupos que debieron generar las conversiones en la representación intermedia, deberán traducir el código de las conversiones a código Assembler.

. Los grupos que deben considerar el chequeo de tipos y/o las conversiones en esta etapa, deberán agregar código para efectuar las conversiones cuando las mismas sean requeridas, siempre que el lenguaje lo permita. Si hubiera incompatibilidad de tipos, esto deberá ser informado por el compilador como un error.

Conversiones explícitas:

. Los grupos cuyos lenguajes permitían conversiones explícitas, deberán traducir el código de las conversiones a código Assembler.

Controles en Tiempo de Ejecución

Incorporar los chequeos en tiempo de ejecución que correspondan al tema particular asignado al grupo**.** Cada grupo deberá efectuar el o los chequeos indicados para situaciones de error que pueden producirse en tiempo de ejecución. El código generado por el compilador deberá, cada vez que se produzca la situación de error correspondiente, emitir un mensaje de error, y finalizar la ejecución.

1. **División por cero:**
   El código Assembler deberá chequear que el divisor sea diferente de cero antes de efectuar una división. En caso que sea cero, deberá emitir un mensaje de error y terminar.
2. **Overflow en sumas / Overflow en productos:**
   El código Assembler deberá controlar el resultado de la operación indicada. Si el mismo excede el rango del tipo del resultado, deberá emitir un mensaje de error y terminar.
3. **Resultados negativos en restas:**
   El código Assembler deberá controlar el resultado de la operación indicada. Este control se aplicará a operaciones entre enteros sin signo. En caso que una resta entre datos de este tipo arroje un resultado negativo, deberá emitir un mensaje de error y terminar.
4. **Pérdida de información en conversiones:**
   El código Assembler deberá controlar que al efectuar una conversión, no haya pérdida de información del dato convertido. En caso que se produzca tal pérdida, se debe emitir un mensaje de error y finalizar la ejecución.

Salidas del compilador

1) Los errores generados en cada una de las etapas (Análisis Léxico, Sintáctico y Generación de Código Intermedio) se deberán mostrar todos juntos, indicando la descripción de cada error y la línea en la que fue detectado.
2) Representación intermedia, según indicaciones del Trabajo Práctico 3.
3) Contenidos de la Tabla de Símbolos.
4) Archivo conteniendo el código Assembler.

Nota: No se deberán mostrar los tokens ni las estructuras sintácticas que se pidieron como salida de los Trabajos Prácticos 1 y 2, a menos que exista una reentrega pendiente que así lo requiera.

Informe

Se debe presentar un informe que incluya:

- **Número de grupo** e integrantes. Colocar direcciones de correo para contacto.
- Introducción
- Descripción de la Generación de Código Intermedio, indicando:
  - estructura utilizada para el almacenamiento del código intermedio,
  - uso de notación posicional de Yacc ($$, $n), indicando en qué casos se usó, y con qué fin,
  - descripción tipo pseudocódigo del algoritmo usado para la generación de las bifurcaciones en sentencias de control,
  - nuevos errores considerados,
  - y todo otro aspecto que se considere relevante.
- Descripción del proceso de Generación de Código Assembler, indicando:
  - Mecanismo utilizado para la generación del código Assembler
  - Mecanismo utilizado para efectuar cada una de las operaciones aritméticas
  - Mecanismo utilizado para la generación de las etiquetas.
  - Todo otro aspecto que se considere relevante
- Modificaciones a las etapas anteriores, si hubieran existido.
- Para los **temas 11, 12 y 13**, incluir una descripción del modo en que fue resuelto el tema correspondiente en cada etapa del desarrollo del compilador.
- Conclusiones

Forma de entrega

Se deberá entregar:

a) Código fuente completo y ejecutable, incluyendo librerías del lenguaje y todo otro componente que fuera necesario para la ejecución.
b) Informe