TEC | Tecnológico de Costa Rica

Instituto Tecnológico de Costa Rica
Área Académica de Ingeniería en Computadores
CE 4301 — Arquitectura de Computadores I

Sesión asincrónica semana 12
Simulador GEM5

Fecha de asignación:	12 octubre 2023	Fecha de entrega:	30 octubre 2023
Grupo:	1 persona	Profesor:	Luis Chavarría Zamora

En este taller se explorará el simulador GEM5, uno de los más útiles para la investigación para construir modelos de procesador y nuevas funciones sobre un código.

1. ¿Qué es GEM5?

GEM5 es un simulador de procesador y nivel de sistema de código abierto. Se utiliza en la investigación académica y en la industria por empresas como ARM Research, AMD Research, Google, Micron, Metempsy, HP y Samsung. GEM5 nació de la fusión de m5 (marco de simulación de CPU) y GEMS (simulador de tiempo de memoria). Entre sus características se tiene lo siguiente:

- Modelos de CPU intercambiables.
- Un modelo de GPU completamente integrado.
- Sistema de memoria basado en eventos.
- Soporte para múltiples ISAs, separando semánticamente del modelo de CPU:
  - Alpha.
  - ARM.
  - SPARC.
  - MIPS.
  - POWER.
  - RISC-V.
  - x86.
- Sistema completamente compatible con:
  - Alpha.
  - ARM.
  - x86.

Una serie de publicaciones han utilizado GEM5. **En este caso, para este taller exploraremos las capacidades para explorar un modelo de CPU.**

2. Instalación de herramientas

Previo a esto se recomienda usar como idioma para el distro de Linux el idioma Inglés, porque la comunidad y las soluciones en Inglés es más grande. También se recomienda tener por lo menos 10 GB libres.

Para instalar la herramienta siga los siguientes pasos:

1. Abra un terminal (puede usar el comando rápido `Ctrl + Alt + T`).
2. Instale los siguientes paquetes:
   - Si tiene la versión de Ubuntu 18.04:

     ```
     sudo apt install build-essential git m4 scons zlib1g zlib1g-dev \
     libprotobuf-dev protobuf-compiler libprotoc-dev python3-dev \
     python3-six python libboost-all-dev libgoogle-perftools-dev \
     swig python-protobuf pkg-config libhdf5-dev libpng-dev
     ```

   - Si tiene la versión de Ubuntu 20.04:

     ```
     sudo apt install build-essential git m4 scons zlib1g zlib1g-dev \
     libprotobuf-dev protobuf-compiler  libgoogle-perftools-dev \
     python3-dev python3-six python-is-python3 pkg-config \
     swig python-protobuf libboost-all-dev libprotoc-dev libhdf5-dev \
     libpng-dev
     ```

3. Es necesario que tenga por lo menos Python 3.X. Instale `pydot`:

   ```
   pip3 install pydot gem5art-artifact gem5art-run gem5art-tasks
   ```

4. Diríjase hacia la carpeta de `Documents` escribiendo el siguiente comando (puede cambiar el nombre si la distribución es en otro idioma):

   ```
   cd Documents/
   ```

5. Ahí se hace una carpeta llamada `GEM5` y entra a la carpeta. Ejecute el siguiente comando:

   ```
   mkdir GEM5 && cd GEM5
   ```

6. Clone el repositorio para construir GEM5:

```
git clone https://gem5.googlesource.com/public/gem5
```

   a) Si no le clona el repositorio puede ser que no tenga instalado `git`, entonces, ejecute el siguiente comando:

```
sudo apt install git-all
```

   b) Si no le funciona clone el siguiente repositorio:

```
git clone git@gitlab.com:abhijitcse/gem5.git
```

7. Se mueve a la carpeta interior:

```
cd gem5/
```

8. Comienza la construcción de GEM5 propiamente (este proceso dependiendo de su computador puede tomar de unos cuantos minutos a una cuantas horas). Se recomienda no manipular el computador mientras se esté ejecutando para que no dure tanto. **Se escogerá un ISA: `X86` y con el BINARY: `gem5.opt`** [1]:

```
scons build/ISA/BINARY -j <Número_de_CPUs_+_1>
```

   - En el espacio de `ISA` puede colocar lo siguiente:
     - `ALPHA`.
     - `ARM`.
     - `MIPS`.
     - `SPARC`.
     - `RISC-V`.
     - `X86`.
     - `NULL`.
   - En el espacio de `BINARY` puede colocar lo siguiente:

[1] **si tiene un problema de bibliotecas revise cómo usar Conda para generar un ambiente virtual independiente sin conflicto**

- `gem5.debug`
- `gem5.opt`
- `gem5.prof`
- `gem5.perf`
- `gem5.fast`

- En cuanto al `<Número_de_CPUs_+_1>` para que dure el menor tiempo posible. Para hacer esto introduzca el siguiente comando en el terminal [2]:

  ```
  cat /proc/cpuinfo | grep processor | wc -l
  ```

  La salida es el número de CPUs disponibles, para el terminal tiene que sumarle 1.
- El comando que ejecuta el profesor es el siguiente (tiene 4 CPUs disponibles):

  ```
  scons build/X86/gem5.opt -j 5
  ```

  Si el sistema no captura la versión de Python como `Python3` puede generar un alias de la siguiente manera:

  ```
  alias scons="/usr/bin/env python3 $(which scons)"
  ```

  Luego de esto vuelve a construir el `scons`.

9. Después de realizar esto pueden ejecutar la siguiente línea de comando:

   ```
   build/X86/gem5.opt configs/example/se.py -c tests/test-progs/hello/bin/x86/linux/hello
   ```

   A grandes rasgos:

   - `gem5.opt`: Es el archivo binario para ejecutar el procesador que construyeron.
   - `se.py`: Es el archivo con la configuración del procesador, usan características como:
     - Tipos de CPU.
     - Niveles de caché.
     - Tamaño de la caché.
     - Asociatividad.
     - Tamaño del bloque.
     - Branch predictor.

[2] este comando usa *pipes*, investigue: ¿qué es?

10. Ahí debe decir al final algo así:

```
**** REAL SIMULATION ****
info: Entering event queue @ 0.  Starting simulation...
Hello world!
Exiting @ tick 5943000 because exiting with last active thread context
```

11. Podemos analizar el archivo de salida al escribir el siguiente comando:

```
gedit m5out/stats.txt
```

Analice las múltiples estadísticas que están a mano. Vemos que esta herramienta nos permite analizar múltiples características de un procesador.

12. Para generar el diagrama de configuración del sistema ejecute el siguiente comando (si no ejecutó el paso 3 no le va a generar la imagen):

```
dot -Tpng -o config.png m5out/config.dot
```

13. Analice el archivo de configuración `se.py`:

```
gedit configs/example/se.py
```

Ahí podemos configurar las características del procesador.

3. Ejecución

Vamos a ejecutar unas cuantas pruebas para comprobar el sistema:

1. Indague los tipos de procesador que están disponibles al ejecutar:

```
build/X86/gem5.opt configs/example/se.py -c tests/test-progs/hello/bin/x86/linux/hello --list-cpu-types
```

2. Se va a ejecutar los siguientes comandos cambiando el tipo de CPU:

```
mkdir out_atomic out_o3 out_timing
build/X86/gem5.opt -d out_timing/ configs/example/se.py -c tests/test-progs/hello/bin/x86/linux/hello --cpu-type=TimingSimpleCPU --caches
build/X86/gem5.opt -d out_o3/ configs/example/se.py -c tests/test-progs/hello/bin/x86/linux/hello --cpu-type=DerivO3CPU --caches
build/X86/gem5.opt -d out_atomic/ configs/example/se.py -c tests/test-progs/hello/bin/x86/linux/hello --cpu-type=AtomicSimpleCPU --caches
```

Se habilitaron las cachés con `--caches`, ahí toman valores por defecto, luego se guarda en las carpetas creadas.

3. Se analizan las diferencias entre uno y otro:

```
diff -u out_o3/stats.txt out_timing/stats.txt &> m5out/diff_o3_timing.txt
diff -u out_o3/stats.txt out_atomic/stats.txt &> m5out/diff_o3_atomic.txt
diff -u out_timing/stats.txt out_atomic/stats.txt &> m5out/diff_timing_atomic.txt
```

4. Si trata de analizar los archivos con `grep` verá que son 533 líneas de resultados, por esta razón es importante saber de antemano qué aspecto va a cambiar.

5. Para observar las características con que se ejecutó el código debe ejecutar el siguiente comando:

```
gedit m5out/config.ini
```

6. Los aspectos que se pueden cambiar en GEM5 se pueden consultar con el siguiente comando:

```
build/X86/gem5.opt -d out_timing/ configs/example/se.py --help
```

7. Otras de las características que se pueden modificar es el *branch predictor* el cual por defecto no está habilitado. Para ver las opciones de *branch predictor* ejecute el siguiente comando:

```
build/X86/gem5.opt -d out_timing/ configs/example/se.py --list-bp-types
build/X86/gem5.opt -d out_timing/ configs/example/se.py --list-indirect-bp-types
```

8. Para definir el *branch predictor* ejecute el siguiente comando (por ejemplo poniendo: `TournamentBP`):

```
build/X86/gem5.opt -d out_timing/ configs/example/se.py -c tests/test-progs/hello/bin/x86/linux/hello --cpu-type=TimingSimpleCPU --caches --bp-type=TournamentBP
```

9. Lo interesante de usar GEM5 es que podemos ejecutar benchmarks como se muestra en la siguiente sección.

4. Uso con benchmark

1. Vaya a la carpeta de *Documents*:

   ```
   cd && cd Documents/
   ```

2. Genere un directorio para trabajar:

   ```
   mkdir spec && cd spec
   ```

3. Clone el repositorio de SPEC:

   ```
   git clone https://github.com/timberjack/Project1_SPEC.git
   ```

4. Vaya a la carpeta recién creada:

   ```
   cd Project1_SPEC
   ```

5. Si escribe `ls` en la línea de comando observará 5 carpetas y 1 script. Vamos a modificar el script para ejecutar el `401.bzip2`. Para esto ejecute las siguientes líneas en el terminal:

   ```
   % Da permisos para ejecutar
   chmod +x runGem5.sh
   % Copia en 401.bzip2
   cp runGem5.sh 401.bzip2/
   cd 401.bzip2/
   ```

6. En el archivo `run.sh` viene un ejemplo de como se debe ejecutar el benchmark. Para observarlo ejecute lo siguiente:

   ```
   cat run.sh
   ```

7. Es necesario modificar el archivo `runGem5.sh` para poder ejecutar el *benchmark*.

```
gedit runGem5.sh
```

8. Solo modifique las líneas 2 y 4 para que coincida con su computador [3]:

```
% Dirección donde está gem5 con el build
export GEM5_DIR=/home/USUARIO/Documents/gem5/gem5
% Ejecución del benchmark según lo observado en el punto anterior
export ARGUMENT=./data/input.program
```

9. Les va a generar un error con el tipo de procesador, para esto reemplace `atomic` con `TimingSimpleCPU` que es un nombre válido. También `-d ~/m5out` por `-d m5out/`

10. El benchmark por defecto duraría días ejecutándose en un computador modesto, para poder ejecutarlo en un período menor es necesario ajustar el código del benchmark. Para conocer cuales archivos del benchmark modificar explore el `Makefile` en `src`. Para este, solo dirijase al `src` y abra el archivo:

```
cd src/
gedit spec.c
```

No siempre el archivo se llama `spec.c`. Para saber cual, explore el `Makefile`.

11. Modifique las líneas 284 y 288 respectivamente con lo siguiente:

```
// Línea 284, disminuye el tamaño de un MB a una porción.
#define MB (1024*1024/32)
// Línea 288, disminuye el tamaño por defecto que descomprime al no
// especificar, pasándolo a 1MB/32. Esto lo hacemos en el .sh.
    int input_size=1, compressed_size;
```

12. Guarde los cambios, cierre el archivo y genere de nuevo el objeto con Make (afortunadamente es x86). Escriba lo siguiente:

```
make
```

[3] revise la ruta en el Terminal para verificar que está bien escrita palabra por palabra

13. Vuelva al directorio para ejecutar el archivo `runGem5.sh`.

    ```
    cd ../
    ```

14. Cierre el editor de texto y proceda a ejecutarlo, puede durar unos cuantos minutos.

    ```
    ./runGem5.sh
    ```

    Sí observa que dura menos de un segundo, como 0.001 segundos hay un error. Revise las rutas proveídas o el modelo de CPU, por defecto viene `atomic`, el cual no existe. El error indica una sugerencia de nombres.

15. Una vez concluye indica los tiempos de simulación en consola. El profesor con su computador de 4 núcleos le duró alrededor de media hora. Puede revisar las estadísticas en la carpeta `m5out`.

5. Ejercicio

En este punto ya usted ejecutó el benchmark `401.bizp2` a cabalidad. Por esta razón se le solicita proponer 3 cambios al benchmark `470.lbm`. Para esto realice lo siguiente:

- Explique en qué consiste el benchmark, y basado en esto, cuales cambios podrían mejorar el rendimiento del sistema. Debe referenciar, sino, no se revisará.
- Ejecute los tres cambios propuestos propiamente del benchmark, solo debe cambiar un parámetro a la vez.
- Si no coincide, con la teoría justifique porqué.
- Guarde los `stats.txt` de cada cambio (deben haber tres al final) y el `runGem5.sh` que generó cada cambio, use un sistema de numeración intuitivo (no usar números grandes poco comprensibles)

6. Entregable

Se debe de subir en la sección de Evaluaciones los siguientes archivos en una carpeta comprimida (`T4_NombreCompleto.zip`):

1. Archivo de reporte en formato PDF.

2. Archivos:

   - `stats#.txt`
   - `runGem5_#.sh`

Si tienen dudas puede escribir al profesor al correo electrónico. **Los documentos serán sometidos a control de plagios**. La entrega se debe realizar por medio del TEC-Digital en la pestaña de evaluación. La fecha de entrega a las 11:59 pm es el tiempo máximo de entrega sin penalidad, después de este tiempo, se bajará un punto por minuto por entrega tardía.