1. **Kurze Beschreibung der einzelnen Klassen**

* **BackendMIPS:**

In BackendMIPS sind die Methoden laut Angabe CA1 implementiert.

* **GlobalPointerRegister:**

Hier werden Bytes in der „Static Data Area“ allociert. Für eine passende Wortlänge befindet sich hier die Methode doWordAlignment.

* **Register:**

So wird ein Register mit Name und Registernummer gespeichert. Abfrage und Setzen ob das Register in Verwendung ist.

* **Registers:**

Hier befindet sich das Registermanagement, wie T und S Register anfordern und freigeben. Abrufen der restlichen Register, wie z.B: GlobalPointerRegister, FramePointerRegister, etc.

* **Segment:**

Abfrage von Data- und Textsegment „String“.

* **StackPointerRegister:**

Hier werden Bytes auf dem Stack allociert, freigegeben und WordAlignment kann vorgenommen werden. Offsets werden Prozeduren-übergreifend verwaltet.

* **SyscallCode:**

Hier befinden sich Syscall Enums. Diese können nun auch mithilfe des Namens und nicht nur mit der Zahl im Backend verwendet werden.

1. **Code Coverage**

* Code Coverage wurde mit Hilfe von IntelliJ IDEA vorgenommen.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Test\_1** | **Test\_2** | **Test\_3** | **Test\_4** | **Test\_5** | **Test\_6** | **Test\_Types** |
| **BackendMIPS** |
| Methodenabdeckung | 49% | 50% | 59% | 61% | 70% | 63% | 75% |
| Zeilenabdeckung | 46% | 45% | 53% | 60% | 69% | 58% | 69% |
| **GlobalPointerRegister** |
| Methodenabdeckung | 100% | 33% | 100% | 100% | 100% | 100% | 100% |
| Zeilenabdeckung | 76% | 23% | 76% | 70% | 70% | 70% | 70% |
| **Register** |
| Methodenabdeckung | 100% | 100% | 100% | 100% | 100% | 100% | 100% |
| Zeilenabdeckung | 100% | 100% | 100% | 100% | 100% | 100% | 100% |
| **Registers** |
| Methodenabdeckung | 68% | 63% | 68% | 68% | 78% | 68% | 73% |
| Zeilenabdeckung | 69% | 68% | 69% | 71% | 75% | 71% | 72% |
| **Segment** |
| Methodenabdeckung | 100% | 100% | 100% | 100% | 100% | 100% | 100% |
| Zeilenabdeckung | 100% | 100% | 100% | 100% | 100% | 100% | 100% |
| **StackPointerRegister** |
| Methodenabdeckung | 83% | 100% | 100% | 100% | 100% | 100% | 100% |
| Zeilenabdeckung | 70% | 91% | 91% | 91% | 91% | 91% | 91% |
| **SyscallCode** |
| Methodenabdeckung | 100% | 100% | 100% | 100% | 100% | 100% | 100% |
| Zeilenabdeckung | 100% | 100% | 100% | 100% | 100% | 100% | 100% |

1. **Code Coverage - Detailliert**

* Von uns wurden zusätzlich zu den Testfällen 1-6 den Testfall „Test\_Types“ hinzugefügt. Mit diesem Testfall sind folgende Methoden zusätzlich abgedeckt worden
  + zeroReg
  + neg
  + div
  + mod
  + isLess
  + isLessOrEqual
  + isEqual
  + not
  + and
  + or
  + branchIf
* Durch unseren zusätzlichen Testfall wurde somit eine Methodenabdeckung von 56 Methoden von insgesamt 56 Methoden erreicht 🡪 100% Methodenabdeckung.

Ohne diesen Testfall waren es 45 Methoden 🡪 80,35% Methodenabdeckung.

Insgesamt konnte also eine Erhöhung von 19,65% erreicht werden.

* Auf eine Zeilenabdeckung von Exceptions und Errors wurde verzichtet, da es wenig Sinn hat, Testfälle nur mit Error-Ausgabe zu schreiben.

**Agenda:**

* Grün – Vollständige Abdeckung der gesamten Methode
* Gelb – Zeilenweise Abdeckung der Methode
* Rot – Keine Abdeckung der Methode

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Test\_1 | Test\_2 | Test\_3 | Test\_4 | Test\_5 | Test\_6 | **Test\_Types** |
| **BackendMIPS** |
| BackendMIPS |  |  |  |  |  |  |  |
| changeSegment |  |  |  |  |  |  |  |
| wordSize |  |  |  |  |  |  |  |
| boolValue |  |  |  |  |  |  |  |
| allocReg |  |  |  |  |  |  |  |
| freeReg |  |  |  |  |  |  |  |
| zeroReg |  |  |  |  |  |  |  |
| comment |  |  |  |  |  |  |  |
| emitLabel |  |  |  |  |  |  |  |
| allocStaticData |  |  |  |  |  |  |  |
| allocStringConstant |  |  |  |  |  |  |  |
| allocStack |  |  |  |  |  |  |  |
| allocHeap |  |  |  |  |  |  |  |
| move |  |  |  |  |  |  |  |
| storeArrayDim |  |  |  |  |  |  |  |
| allocArray |  |  |  |  |  |  |  |
| loadConst |  |  |  |  |  |  |  |
| loadAdress |  |  |  |  |  |  |  |
| loadWord |  |  |  |  |  |  |  |
| storeWord |  |  |  |  |  |  |  |
| loadWordReg |  |  |  |  |  |  |  |
| loadWordReg |  |  |  |  |  |  |  |
| storeWordReg |  |  |  |  |  |  |  |
| arrayOffset |  |  |  |  |  |  |  |
| arrayLength |  |  |  |  |  |  |  |
| writeString |  |  |  |  |  |  |  |
| syscall |  |  |  |  |  |  |  |
| neg |  |  |  |  |  |  |  |
| add |  |  |  |  |  |  |  |
| addConst |  |  |  |  |  |  |  |
| sub |  |  |  |  |  |  |  |
| mul |  |  |  |  |  |  |  |
| mulConst |  |  |  |  |  |  |  |
| div |  |  |  |  |  |  |  |
| mod |  |  |  |  |  |  |  |
| isLess |  |  |  |  |  |  |  |
| isLessOrEqual |  |  |  |  |  |  |  |
| isEqual |  |  |  |  |  |  |  |
| not |  |  |  |  |  |  |  |
| and |  |  |  |  |  |  |  |
| or |  |  |  |  |  |  |  |
| branchIf |  |  |  |  |  |  |  |
| jump |  |  |  |  |  |  |  |
| enterMain |  |  |  |  |  |  |  |
| exitMain |  |  |  |  |  |  |  |
| enterProc |  |  |  |  |  |  |  |
| exitProc |  |  |  |  |  |  |  |
| jumpRegister |  |  |  |  |  |  |  |
| returnFromProc |  |  |  |  |  |  |  |
| prepareProcCall |  |  |  |  |  |  |  |
| jumpAndLink |  |  |  |  |  |  |  |
| passArg |  |  |  |  |  |  |  |
| callProc |  |  |  |  |  |  |  |
| paramOffset |  |  |  |  |  |  |  |
| writePredefinedProcedures |  |  |  |  |  |  |  |
| writeProcedure\_writeint |  |  |  |  |  |  |  |