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**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Testfälle** | **1** | **2** | **3** | **4** | **5** | **6** | **Error** | **IfElse** | **Types** | **If\_Exception\_**  **GlobalPointerReg** | **Summe** |
| **BackendMIPS** |
| Methodenabdeckung | 49% | 50% | 59% | 61% | 70% | 63% | 3% | 47% | 64% | 47% | 100% |
| Zeilenabdeckung | 42% | 42% | 51% | 57% | 66% | 55% | 2% | 39% | 57% | 41% | 100% |
| **GlobalPointerRegister** |
| Methodenabdeckung | 100% | 33% | 100% | 100% | 100% | 100% | 33% | 33% | 33% | 100% | 100% |
| Zeilenabdeckung | 75% | 23% | 76% | 70% | 70% | 70% | 23% | 23% | 23% | 82% | 100% |
| **Register** |
| Methodenabdeckung | 100% | 100% | 100% | 100% | 100% | 100% | 33% | 100% | 100% | 100% | 100% |
| Zeilenabdeckung | 100% | 100% | 100% | 100% | 100% | 100% | 58% | 100% | 100% | 100% | 100% |
| **Registers** |
| Methodenabdeckung | 64% | 58% | 64% | 64% | 76% | 64% | 5% | 58% | 64% | 58% | 76% |
| Zeilenabdeckung | 63% | 63% | 64% | 67% | 71% | 67% | 35% | 64% | 67% | 57% | 67% |
| **Segment** |
| Methodenabdeckung | 100% | 100% | 100% | 100% | 100% | 100% | 0% | 100% | 100% | 100% | 100% |
| Zeilenabdeckung | 100% | 100% | 100% | 100% | 100% | 100% | 0% | 100% | 100% | 100% | 100% |
| **StackPointerRegister** |
| Methodenabdeckung | 83% | 100% | 100% | 100% | 100% | 100% | 16% | 83% | 83% | 83% | 100% |
| Zeilenabdeckung | 70% | 91% | 91% | 91% | 91% | 91% | 12% | 70% | 70% | 70% | 91% |
| **SyscallCode** |
| Methodenabdeckung | 100% | 100% | 100% | 100% | 100% | 100% | 0% | 100% | 100% | 100% | 100% |
| Zeilenabdeckung | 100% | 100% | 100% | 100% | 100% | 100% | 0% | 100% | 100% | 100% | 100% |

1. **Code Coverage - Detailliert**

**Agenda:**

* Grün – Vollständige Abdeckung der gesamten Methode
* Gelb – Zeilenweise Abdeckung der Methode
* Rot – Keine Abdeckung der Methode
* Von uns wurden zusätzlich zu den Testfällen 1-6 die Testfälle „**Test\_fak**“, „**Test\_Error**“, „**Test\_IfElse**“, „**Test\_Types**“ hinzugefügt, durch welche die Methodenabdeckung, als auch die Zeilenabdeckung der Klasse **BackendMIPS** zu 100% gegeben ist.
* Mit dem Testfall **„Test\_If\_Exception”** wurde der Rest der Klasse GlobalPointerRegister abgedeckt.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Testfälle | 1 | 2 | 3 | 4 | 5 | 6 | Error | Types | IfElse | If\_Exception\_ GlobalPointerReg | In Summe |
| **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 |  |  |  |  |  |  |  |  |  |  |  |
| **GlobalPointerRegister** |
| GlobalPointerRegister |  |  |  |  |  |  |  |  |  |  |  |
| allocateBytes |  |  |  |  |  |  |  |  |  |  |  |
| doWordAlignment |  |  |  |  |  |  |  |  |  |  |  |
| **Register** |
| Register |  |  |  |  |  |  |  |  |  |  |  |
| Register |  |  |  |  |  |  |  |  |  |  |  |
| getName |  |  |  |  |  |  |  |  |  |  |  |
| getRegisterNumber |  |  |  |  |  |  |  |  |  |  |  |
| isInUse |  |  |  |  |  |  |  |  |  |  |  |
| setInUse |  |  |  |  |  |  |  |  |  |  |  |
| **Registers** |
| Registers |  |  |  |  |  |  |  |  |  |  |  |
| getZeroRegister |  |  |  |  |  |  |  |  |  |  |  |
| getStackPointerRegister |  |  |  |  |  |  |  |  |  |  |  |
| getGlobalPointerRegister |  |  |  |  |  |  |  |  |  |  |  |
| getFramePointerRegister |  |  |  |  |  |  |  |  |  |  |  |
| getReturnAdressRegister |  |  |  |  |  |  |  |  |  |  |  |
| getUnusedRegister |  |  |  |  |  |  |  |  |  |  |  |
| getRegisterByNumber |  |  |  |  |  |  |  |  |  |  |  |
| freeRegister |  |  |  |  |  |  |  |  |  |  |  |
| freeAllRegister |  |  |  |  |  |  |  |  |  |  |  |
| getV0 |  |  |  |  |  |  |  |  |  |  |  |
| getV1 |  |  |  |  |  |  |  |  |  |  |  |
| getA0 |  |  |  |  |  |  |  |  |  |  |  |
| getA1 |  |  |  |  |  |  |  |  |  |  |  |
| getA2 |  |  |  |  |  |  |  |  |  |  |  |
| getA3 |  |  |  |  |  |  |  |  |  |  |  |
| getArrayLengthRegister |  |  |  |  |  |  |  |  |  |  |  |
| **Segment** |
| Segment |  |  |  |  |  |  |  |  |  |  |  |
| toString |  |  |  |  |  |  |  |  |  |  |  |
| **StackPointerRegister** |
| StackPointerRegister |  |  |  |  |  |  |  |  |  |  |  |
| allocateBytes |  |  |  |  |  |  |  |  |  |  |  |
| doWordAlignment |  |  |  |  |  |  |  |  |  |  |  |
| freeBytes |  |  |  |  |  |  |  |  |  |  |  |
| requestNewOffset |  |  |  |  |  |  |  |  |  |  |  |
| deleteCurrentOffset |  |  |  |  |  |  |  |  |  |  |  |
| **SyscallCode** |
| SyscallCode |  |  |  |  |  |  |  |  |  |  |  |
| getValue |  |  |  |  |  |  |  |  |  |  |  |