

MToy

1.0

Generated by Doxygen 1.8.17

1 MIPS Simulator	1
1.1 Introduction	1
1.2 Language Support	2
1.3 Syscall List	2
1.4 Build	2
1.4.1 Build Environment	2
1.4.2 Preparation	2
1.4.3 Compilation	3
1.4.4 Special Notice	3
2 The Design Brochure	5
2.1 The Procedure of Simulation	5
2.2 The design of MMAP/MUNMAP Simulation	5
2.3 SigFault Capturing	5
2.4 Code Deduplication	5
2.5 Store the Instructions	5
3 Todo List	7
4 Namespace Index	9
4.1 Namespace List	9
5 Hierarchical Index	11
5.1 Class Hierarchy	11
6 Class Index	13
6.1 Class List	13
7 File Index	15
7.1 File List	15
8 Namespace Documentation	17
8.1 _SIM Namespace Reference	17
8.1.1 Detailed Description	17
8.1.2 Typedef Documentation	17
8.1.2.1 InstrPtr	17
8.1.3 Function Documentation	18
8.1.3.1 make_unique()	18
8.2 Ui Namespace Reference	19
8.2.1 Detailed Description	19
9 Class Documentation	21
9.1 ADDImpl Struct Reference	21
9.1.1 Constructor & Destructor Documentation	21
9.1.1.1 ADDImpl()	21

9.1.2 Member Function Documentation	22
9.1.2.1 exec()	22
9.2 ADDImpl Struct Reference	22
9.2.1 Constructor & Destructor Documentation	22
9.2.1.1 ADDImpl()	22
9.2.2 Member Function Documentation	23
9.2.2.1 exec()	23
9.3 ADDIImpl Struct Reference	23
9.3.1 Constructor & Destructor Documentation	23
9.3.1.1 ADDIImpl()	23
9.3.2 Member Function Documentation	24
9.3.2.1 exec()	24
9.4 ADDUImpl Struct Reference	24
9.4.1 Constructor & Destructor Documentation	24
9.4.1.1 ADDUImpl()	24
9.4.2 Member Function Documentation	25
9.4.2.1 exec()	25
9.5 ANDImpl Struct Reference	25
9.5.1 Constructor & Destructor Documentation	25
9.5.1.1 ANDImpl()	25
9.5.2 Member Function Documentation	26
9.5.2.1 exec()	26
9.6 ANDIImpl Struct Reference	26
9.6.1 Constructor & Destructor Documentation	26
9.6.1.1 ANDIImpl()	26
9.6.2 Member Function Documentation	27
9.6.2.1 exec()	27
9.7 BEQImpl Struct Reference	27
9.7.1 Constructor & Destructor Documentation	27
9.7.1.1 BEQImpl()	27
9.7.2 Member Function Documentation	28
9.7.2.1 exec()	28
9.8 BGEZALImpl Struct Reference	28
9.8.1 Constructor & Destructor Documentation	28
9.8.1.1 BGEZALImpl()	28
9.8.2 Member Function Documentation	29
9.8.2.1 exec()	29
9.9 BGEZALLImpl Struct Reference	29
9.9.1 Constructor & Destructor Documentation	29
9.9.1.1 BGEZALLImpl()	29
9.9.2 Member Function Documentation	30
9.9.2.1 exec()	30

9.10 BGEZImpl Struct Reference	30
9.10.1 Constructor & Destructor Documentation	30
9.10.1.1 BGEZImpl()	30
9.10.2 Member Function Documentation	31
9.10.2.1 exec()	31
9.11 BGEZLImpl Struct Reference	31
9.11.1 Constructor & Destructor Documentation	31
9.11.1.1 BGEZLImpl()	31
9.11.2 Member Function Documentation	32
9.11.2.1 exec()	32
9.12 BGTZImpl Struct Reference	32
9.12.1 Constructor & Destructor Documentation	32
9.12.1.1 BGTZImpl()	32
9.12.2 Member Function Documentation	33
9.12.2.1 exec()	33
9.13 BLEZImpl Struct Reference	33
9.13.1 Constructor & Destructor Documentation	33
9.13.1.1 BLEZImpl()	33
9.13.2 Member Function Documentation	34
9.13.2.1 exec()	34
9.14 BLTZALImpl Struct Reference	34
9.14.1 Constructor & Destructor Documentation	34
9.14.1.1 BLTZALImpl()	34
9.14.2 Member Function Documentation	35
9.14.2.1 exec()	35
9.15 BLTZALLImpl Struct Reference	35
9.15.1 Constructor & Destructor Documentation	35
9.15.1.1 BLTZALLImpl()	35
9.15.2 Member Function Documentation	36
9.15.2.1 exec()	36
9.16 BLTZImpl Struct Reference	36
9.16.1 Constructor & Destructor Documentation	36
9.16.1.1 BLTZImpl()	36
9.16.2 Member Function Documentation	37
9.16.2.1 exec()	37
9.17 BLTZLImpl Struct Reference	37
9.17.1 Constructor & Destructor Documentation	37
9.17.1.1 BLTZLImpl()	37
9.17.2 Member Function Documentation	38
9.17.2.1 exec()	38
9.18 BNEImpl Struct Reference	38
9.18.1 Constructor & Destructor Documentation	38

9.18.1.1 BNEImpl()	38
9.18.2 Member Function Documentation	39
9.18.2.1 exec()	39
9.19 BREAKImpl Struct Reference	39
9.19.1 Constructor & Destructor Documentation	39
9.19.1.1 BREAKImpl()	39
9.19.2 Member Function Documentation	40
9.19.2.1 exec()	40
9.20 CLOImpl Struct Reference	40
9.20.1 Constructor & Destructor Documentation	40
9.20.1.1 CLOImpl()	40
9.20.2 Member Function Documentation	41
9.20.2.1 exec()	41
9.21 CLZImpl Struct Reference	41
9.21.1 Constructor & Destructor Documentation	41
9.21.1.1 CLZImpl()	41
9.21.2 Member Function Documentation	42
9.21.2.1 exec()	42
9.22 DIVImpl Struct Reference	42
9.22.1 Constructor & Destructor Documentation	42
9.22.1.1 DIVImpl()	42
9.22.2 Member Function Documentation	43
9.22.2.1 exec()	43
9.23 DIVUImpl Struct Reference	43
9.23.1 Constructor & Destructor Documentation	43
9.23.1.1 DIVUImpl()	43
9.23.2 Member Function Documentation	44
9.23.2.1 exec()	44
9.24 Executor Class Reference	44
9.24.1 Detailed Description	45
9.24.2 Member Function Documentation	45
9.24.2.1 exit	45
9.24.2.2 finished	45
9.24.2.3 next	45
9.24.3 Member Data Documentation	45
9.24.3.1 impls	46
9.24.3.2 mainW	46
9.25 Heap Class Reference	46
9.25.1 Detailed Description	46
9.25.2 Constructor & Destructor Documentation	47
9.25.2.1 ~Heap()	47
9.25.3 Member Function Documentation	47

9.25.3.1 alloc()	47
9.25.3.2 clear()	47
9.25.3.3 dealloc()	47
9.25.3.4 order()	48
9.25.4 Member Data Documentation	48
9.25.4.1 mapping	48
9.25.4.2 size	48
9.26 _SIM::InstrDeleter Struct Reference	48
9.26.1 Detailed Description	49
9.26.2 Member Function Documentation	49
9.26.2.1 operator()	49
9.27 InstructionImpl Struct Reference	49
9.27.1 Detailed Description	50
9.27.2 Constructor & Destructor Documentation	50
9.27.2.1 InstructionImpl()	50
9.27.3 Member Function Documentation	50
9.27.3.1 exec()	51
9.27.4 Member Data Documentation	51
9.27.4.1 instr	51
9.27.4.2 mainW	51
9.28 JALImpl Struct Reference	52
9.28.1 Constructor & Destructor Documentation	52
9.28.1.1 JALImpl()	52
9.28.2 Member Function Documentation	52
9.28.2.1 exec()	52
9.29 JALRImpl Struct Reference	53
9.29.1 Constructor & Destructor Documentation	53
9.29.1.1 JALRImpl()	53
9.29.2 Member Function Documentation	54
9.29.2.1 exec()	54
9.30 JImpl Struct Reference	54
9.30.1 Constructor & Destructor Documentation	54
9.30.1.1 JImpl()	54
9.30.2 Member Function Documentation	55
9.30.2.1 exec()	55
9.31 JRImpl Struct Reference	55
9.31.1 Constructor & Destructor Documentation	56
9.31.1.1 JRImpl()	56
9.31.2 Member Function Documentation	57
9.31.2.1 exec()	57
9.32 LBImpl Struct Reference	57
9.32.1 Constructor & Destructor Documentation	57

9.32.1.1 LBImpl()	57
9.32.2 Member Function Documentation	58
9.32.2.1 exec()	58
9.33 LBUImpl Struct Reference	58
9.33.1 Constructor & Destructor Documentation	58
9.33.1.1 LBUImpl()	58
9.33.2 Member Function Documentation	59
9.33.2.1 exec()	59
9.34 LHImpl Struct Reference	59
9.34.1 Constructor & Destructor Documentation	59
9.34.1.1 LHImpl()	59
9.34.2 Member Function Documentation	60
9.34.2.1 exec()	60
9.35 LHUImpl Struct Reference	60
9.35.1 Constructor & Destructor Documentation	60
9.35.1.1 LHUImpl()	60
9.35.2 Member Function Documentation	61
9.35.2.1 exec()	61
9.36 LLImpl Struct Reference	61
9.36.1 Constructor & Destructor Documentation	61
9.36.1.1 LLImpl()	61
9.36.2 Member Function Documentation	62
9.36.2.1 exec()	62
9.37 LUImpl Struct Reference	62
9.37.1 Constructor & Destructor Documentation	62
9.37.1.1 LUImpl()	62
9.37.2 Member Function Documentation	63
9.37.2.1 exec()	63
9.38 LWImpl Struct Reference	63
9.38.1 Constructor & Destructor Documentation	63
9.38.1.1 LWImpl()	63
9.38.2 Member Function Documentation	64
9.38.2.1 exec()	64
9.39 LWLImpl Struct Reference	64
9.39.1 Constructor & Destructor Documentation	64
9.39.1.1 LWLImpl()	64
9.39.2 Member Function Documentation	65
9.39.2.1 exec()	65
9.40 LWRImpl Struct Reference	65
9.40.1 Constructor & Destructor Documentation	65
9.40.1.1 LWRImpl()	65
9.40.2 Member Function Documentation	66

9.40.2.1 exec()	66
9.41 MADDImpl Struct Reference	66
9.41.1 Constructor & Destructor Documentation	66
9.41.1.1 MADDImpl()	66
9.41.2 Member Function Documentation	67
9.41.2.1 exec()	67
9.42 MADDUImpl Struct Reference	67
9.42.1 Constructor & Destructor Documentation	67
9.42.1.1 MADDUImpl()	67
9.42.2 Member Function Documentation	68
9.42.2.1 exec()	68
9.43 MainWindow Class Reference	68
9.43.1 Detailed Description	70
9.43.2 Constructor & Destructor Documentation	70
9.43.2.1 MainWindow()	70
9.43.2.2 ~MainWindow()	71
9.43.3 Member Function Documentation	71
9.43.3.1 allocHeap()	71
9.43.3.2 deallocHeap()	71
9.43.3.3 decreaseStack()	72
9.43.3.4 edit()	72
9.43.3.5 editHeap()	72
9.43.3.6 editStack()	73
9.43.3.7 fetchHeap()	73
9.43.3.8 fetchStack()	74
9.43.3.9 getRealAddr()	74
9.43.3.10 handleSyscall()	74
9.43.3.11 increaseStack()	75
9.43.3.12 memoryType()	75
9.43.3.13 on_aboutButton_clicked	75
9.43.3.14 on_executeButton_clicked	75
9.43.3.15 on_openButton_clicked	76
9.43.3.16 on_pushButton_clicked	76
9.43.3.17 on_resetButton_clicked	76
9.43.3.18 on_stepButton_clicked	76
9.43.3.19 on_stopButton_clicked	76
9.43.3.20 on_translateButton_clicked	77
9.43.3.21 resetAll()	77
9.43.3.22 showWarning()	77
9.43.3.23 translateAll()	77
9.43.3.24 updateAcc()	77
9.43.3.25 updateHigh()	78

9.43.3.26 updateLow()	78
9.43.3.27 updateProgramCounter()	78
9.43.3.28 updateRegValue()	79
9.43.3.29 updateStack()	79
9.43.4 Member Data Documentation	79
9.43.4.1 ACC	79
9.43.4.2 advanceCounter	79
9.43.4.3 all	80
9.43.4.4 executor	80
9.43.4.5 frame	80
9.43.4.6 heap	80
9.43.4.7 high	80
9.43.4.8 instructions	80
9.43.4.9 low	81
9.43.4.10 part	81
9.43.4.11 PC	81
9.43.4.12 REGS	81
9.43.4.13 stack	81
9.43.4.14 timer	81
9.43.4.15 ui	82
9.44 MFHImpl Struct Reference	82
9.44.1 Constructor & Destructor Documentation	82
9.44.1.1 MFHImpl()	82
9.44.2 Member Function Documentation	83
9.44.2.1 exec()	83
9.45 MFLOImpl Struct Reference	83
9.45.1 Constructor & Destructor Documentation	83
9.45.1.1 MFLOImpl()	83
9.45.2 Member Function Documentation	84
9.45.2.1 exec()	84
9.46 MSUBImpl Struct Reference	84
9.46.1 Constructor & Destructor Documentation	84
9.46.1.1 MSUBImpl()	84
9.46.2 Member Function Documentation	85
9.46.2.1 exec()	85
9.47 MSUBUImpl Struct Reference	85
9.47.1 Constructor & Destructor Documentation	85
9.47.1.1 MSUBUImpl()	85
9.47.2 Member Function Documentation	86
9.47.2.1 exec()	86
9.48 MTHImpl Struct Reference	86
9.48.1 Constructor & Destructor Documentation	86

9.48.1.1 MTHImpl()	86
9.48.2 Member Function Documentation	87
9.48.2.1 exec()	87
9.49 MTLOImpl Struct Reference	87
9.49.1 Constructor & Destructor Documentation	87
9.49.1.1 MTLOImpl()	87
9.49.2 Member Function Documentation	88
9.49.2.1 exec()	88
9.50 MULImpl Struct Reference	88
9.50.1 Constructor & Destructor Documentation	88
9.50.1.1 MULImpl()	88
9.50.2 Member Function Documentation	89
9.50.2.1 exec()	89
9.51 MULTImpl Struct Reference	89
9.51.1 Constructor & Destructor Documentation	89
9.51.1.1 MULTImpl()	89
9.51.2 Member Function Documentation	90
9.51.2.1 exec()	90
9.52 MULTUImpl Struct Reference	90
9.52.1 Constructor & Destructor Documentation	90
9.52.1.1 MULTUImpl()	90
9.52.2 Member Function Documentation	91
9.52.2.1 exec()	91
9.53 NOPIImpl Struct Reference	91
9.53.1 Constructor & Destructor Documentation	91
9.53.1.1 NOPIImpl()	91
9.53.2 Member Function Documentation	92
9.53.2.1 exec()	92
9.54 NORImpl Struct Reference	92
9.54.1 Constructor & Destructor Documentation	92
9.54.1.1 NORImpl()	92
9.54.2 Member Function Documentation	93
9.54.2.1 exec()	93
9.55 ORImpl Struct Reference	93
9.55.1 Constructor & Destructor Documentation	93
9.55.1.1 ORImpl()	93
9.55.2 Member Function Documentation	94
9.55.2.1 exec()	94
9.56 ORImpl Struct Reference	94
9.56.1 Constructor & Destructor Documentation	94
9.56.1.1 ORImpl()	94
9.56.2 Member Function Documentation	95

9.56.2.1 exec()	95
9.57 SBImpl Struct Reference	95
9.57.1 Constructor & Destructor Documentation	95
9.57.1.1 SBImpl()	95
9.57.2 Member Function Documentation	96
9.57.2.1 exec()	96
9.58 SCImpl Struct Reference	96
9.58.1 Constructor & Destructor Documentation	96
9.58.1.1 SCImpl()	96
9.58.2 Member Function Documentation	97
9.58.2.1 exec()	97
9.59 SHImpl Struct Reference	97
9.59.1 Constructor & Destructor Documentation	97
9.59.1.1 SHImpl()	97
9.59.2 Member Function Documentation	98
9.59.2.1 exec()	98
9.60 SLLImpl Struct Reference	98
9.60.1 Constructor & Destructor Documentation	98
9.60.1.1 SLLImpl()	98
9.60.2 Member Function Documentation	99
9.60.2.1 exec()	99
9.61 SLLVImpl Struct Reference	99
9.61.1 Constructor & Destructor Documentation	99
9.61.1.1 SLLVImpl()	99
9.61.2 Member Function Documentation	100
9.61.2.1 exec()	100
9.62 SLTImpl Struct Reference	100
9.62.1 Constructor & Destructor Documentation	100
9.62.1.1 SLTImpl()	100
9.62.2 Member Function Documentation	101
9.62.2.1 exec()	101
9.63 SLTImpl Struct Reference	101
9.63.1 Constructor & Destructor Documentation	101
9.63.1.1 SLTImpl()	101
9.63.2 Member Function Documentation	102
9.63.2.1 exec()	102
9.64 SLTIUImpl Struct Reference	102
9.64.1 Constructor & Destructor Documentation	102
9.64.1.1 SLTIUImpl()	102
9.64.2 Member Function Documentation	103
9.64.2.1 exec()	103
9.65 SLTUImpl Struct Reference	103

9.65.1 Constructor & Destructor Documentation	103
9.65.1.1 SLTImpl()	103
9.65.2 Member Function Documentation	104
9.65.2.1 exec()	104
9.66 SRAImpl Struct Reference	104
9.66.1 Constructor & Destructor Documentation	104
9.66.1.1 SRAImpl()	104
9.66.2 Member Function Documentation	105
9.66.2.1 exec()	105
9.67 SRAVImpl Struct Reference	105
9.67.1 Constructor & Destructor Documentation	105
9.67.1.1 SRAVImpl()	105
9.67.2 Member Function Documentation	106
9.67.2.1 exec()	106
9.68 SRLImpl Struct Reference	106
9.68.1 Constructor & Destructor Documentation	106
9.68.1.1 SRLImpl()	106
9.68.2 Member Function Documentation	107
9.68.2.1 exec()	107
9.69 SRLVImpl Struct Reference	107
9.69.1 Constructor & Destructor Documentation	107
9.69.1.1 SRLVImpl()	107
9.69.2 Member Function Documentation	108
9.69.2.1 exec()	108
9.70 Stack Struct Reference	108
9.70.1 Detailed Description	109
9.70.2 Constructor & Destructor Documentation	109
9.70.2.1 Stack()	109
9.70.2.2 ~Stack()	109
9.70.3 Member Function Documentation	109
9.70.3.1 clear()	109
9.70.3.2 decrease()	109
9.70.3.3 enlarge()	110
9.70.3.4 get()	110
9.70.3.5 grow()	110
9.70.3.6 isEnoughFor()	111
9.70.3.7 order()	111
9.70.3.8 size()	111
9.70.4 Member Data Documentation	111
9.70.4.1 capacity	112
9.70.4.2 current	112
9.70.4.3 highest	112

9.71 SUBImpl Struct Reference	112
9.71.1 Constructor & Destructor Documentation	112
9.71.1.1 SUBImpl()	112
9.71.2 Member Function Documentation	113
9.71.2.1 exec()	113
9.72 SUBUImpl Struct Reference	113
9.72.1 Constructor & Destructor Documentation	113
9.72.1.1 SUBUImpl()	113
9.72.2 Member Function Documentation	114
9.72.2.1 exec()	114
9.73 SWCLImpl Struct Reference	114
9.73.1 Constructor & Destructor Documentation	114
9.73.1.1 SWCLImpl()	114
9.73.2 Member Function Documentation	115
9.73.2.1 exec()	115
9.74 SWImpl Struct Reference	115
9.74.1 Constructor & Destructor Documentation	115
9.74.1.1 SWImpl()	115
9.74.2 Member Function Documentation	116
9.74.2.1 exec()	116
9.75 SWLImpl Struct Reference	116
9.75.1 Constructor & Destructor Documentation	116
9.75.1.1 SWLImpl()	116
9.75.2 Member Function Documentation	117
9.75.2.1 exec()	117
9.76 SWRImpl Struct Reference	117
9.76.1 Constructor & Destructor Documentation	117
9.76.1.1 SWRImpl()	117
9.76.2 Member Function Documentation	118
9.76.2.1 exec()	118
9.77 SYSCALLImpl Struct Reference	118
9.77.1 Constructor & Destructor Documentation	118
9.77.1.1 SYSCALLImpl()	118
9.77.2 Member Function Documentation	119
9.77.2.1 exec()	119
9.78 TEQImpl Struct Reference	119
9.78.1 Constructor & Destructor Documentation	119
9.78.1.1 TEQImpl()	119
9.78.2 Member Function Documentation	120
9.78.2.1 exec()	120
9.79 TEQImpl Struct Reference	120
9.79.1 Constructor & Destructor Documentation	120

9.79.1.1 TEQImpl()	120
9.79.2 Member Function Documentation	121
9.79.2.1 exec()	121
9.80 TGEImpl Struct Reference	121
9.80.1 Constructor & Destructor Documentation	121
9.80.1.1 TGEImpl()	121
9.80.2 Member Function Documentation	122
9.80.2.1 exec()	122
9.81 TGEImpl Struct Reference	122
9.81.1 Constructor & Destructor Documentation	122
9.81.1.1 TGEImpl()	122
9.81.2 Member Function Documentation	123
9.81.2.1 exec()	123
9.82 TGEIImpl Struct Reference	123
9.82.1 Constructor & Destructor Documentation	123
9.82.1.1 TGEIImpl()	123
9.82.2 Member Function Documentation	124
9.82.2.1 exec()	124
9.83 TGEUImpl Struct Reference	124
9.83.1 Constructor & Destructor Documentation	124
9.83.1.1 TGEUImpl()	124
9.83.2 Member Function Documentation	125
9.83.2.1 exec()	125
9.84 TLTIImpl Struct Reference	125
9.84.1 Constructor & Destructor Documentation	125
9.84.1.1 TLTIImpl()	125
9.84.2 Member Function Documentation	126
9.84.2.1 exec()	126
9.85 TLTIImpl Struct Reference	126
9.85.1 Constructor & Destructor Documentation	126
9.85.1.1 TLTIImpl()	126
9.85.2 Member Function Documentation	127
9.85.2.1 exec()	127
9.86 TLTIUImpl Struct Reference	127
9.86.1 Constructor & Destructor Documentation	127
9.86.1.1 TLTIUImpl()	127
9.86.2 Member Function Documentation	128
9.86.2.1 exec()	128
9.87 TLTIUImpl Struct Reference	128
9.87.1 Constructor & Destructor Documentation	128
9.87.1.1 TLTIUImpl()	128
9.87.2 Member Function Documentation	129

9.87.2.1 exec()	129
9.88 TNEImpl Struct Reference	129
9.88.1 Constructor & Destructor Documentation	129
9.88.1.1 TNEImpl()	129
9.88.2 Member Function Documentation	130
9.88.2.1 exec()	130
9.89 TNEImpl Struct Reference	130
9.89.1 Constructor & Destructor Documentation	130
9.89.1.1 TNEImpl()	130
9.89.2 Member Function Documentation	131
9.89.2.1 exec()	131
9.90 XORImpl Struct Reference	131
9.90.1 Constructor & Destructor Documentation	131
9.90.1.1 XORImpl()	131
9.90.2 Member Function Documentation	132
9.90.2.1 exec()	132
9.91 XORImpl Struct Reference	132
9.91.1 Constructor & Destructor Documentation	132
9.91.1.1 XORImpl()	132
9.91.2 Member Function Documentation	133
9.91.2.1 exec()	133
10 File Documentation	135
10.1 README.md File Reference	135
10.2 REPORT.md File Reference	135
10.3 src/executor.cpp File Reference	135
10.4 src/executor.h File Reference	135
10.5 src/fs.h File Reference	135
10.6 src/global.cpp File Reference	136
10.6.1 Variable Documentation	136
10.6.1.1 REG_NAME	136
10.6.1.2 STATIC_HIGH	136
10.7 src/global.h File Reference	136
10.7.1 Macro Definition Documentation	137
10.7.1.1 BASE_ADDR	137
10.7.1.2 LIKELY	137
10.7.1.3 STATIC_LOW	138
10.7.1.4 UNLIKELY	138
10.7.2 Variable Documentation	138
10.7.2.1 REG_NAME	138
10.7.2.2 STATIC_HIGH	138
10.8 src/heap.cpp File Reference	138

10.8.1 Function Documentation	138
10.8.1.1 bind_sigsegy()	139
10.8.1.2 segfault_sigaction()	139
10.9 src/heap.h File Reference	139
10.9.1 Typedef Documentation	140
10.9.1.1 TreeSet	140
10.9.2 Function Documentation	140
10.9.2.1 bind_sigsegy()	140
10.9.2.2 segfault_sigaction()	140
10.10 src/instruction.cpp File Reference	140
10.10.1 Function Documentation	141
10.10.1.1 resolv_type()	141
10.11 src/instruction.h File Reference	141
10.11.1 Macro Definition Documentation	143
10.11.1.1 FCR_ADD	143
10.11.1.2 FCR_ADDU	143
10.11.1.3 FCR_AND	143
10.11.1.4 FCR_BREAK	143
10.11.1.5 FCR_DIV	143
10.11.1.6 FCR_DIVU	144
10.11.1.7 FCR_JALR	144
10.11.1.8 FCR_JR	144
10.11.1.9 FCR_MFHI	144
10.11.1.10 FCR_MFLO	144
10.11.1.11 FCR_MTHI	144
10.11.1.12 FCR_MTLO	144
10.11.1.13 FCR_MULT	144
10.11.1.14 FCR_MULTU	145
10.11.1.15 FCR_NOR	145
10.11.1.16 FCR_OR	145
10.11.1.17 FCR_SLL	145
10.11.1.18 FCR_SLLV	145
10.11.1.19 FCR_SLT	145
10.11.1.20 FCR_SLTU	145
10.11.1.21 FCR_SRA	145
10.11.1.22 FCR_SRAV	146
10.11.1.23 FCR_SRL	146
10.11.1.24 FCR_SRLV	146
10.11.1.25 FCR_SUB	146
10.11.1.26 FCR_SUBU	146
10.11.1.27 FCR_SYSCALL	146
10.11.1.28 FCR_TEQ	146

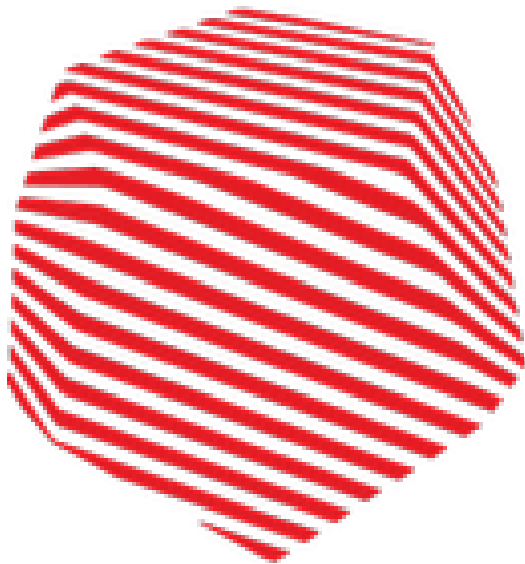
10.11.1.29 FCR_TGE	146
10.11.1.30 FCR_TGEU	147
10.11.1.31 FCR_TLT	147
10.11.1.32 FCR_TLTU	147
10.11.1.33 FCR_TNE	147
10.11.1.34 FCR_XOR	147
10.11.1.35 OPC_ADDI	147
10.11.1.36 OPC_ADDIU	147
10.11.1.37 OPC_ANDI	147
10.11.1.38 OPC_BEQ	148
10.11.1.39 OPC_BGTZ	148
10.11.1.40 OPC_BLEZ	148
10.11.1.41 OPC_BNE	148
10.11.1.42 OPC_J	148
10.11.1.43 OPC_JAL	148
10.11.1.44 OPC_LB	148
10.11.1.45 OPC_LBU	148
10.11.1.46 OPC_LH	149
10.11.1.47 OPC_LHU	149
10.11.1.48 OPC_LL	149
10.11.1.49 OPC_LUI	149
10.11.1.50 OPC_LW	149
10.11.1.51 OPC_LWL	149
10.11.1.52 OPC_LWR	149
10.11.1.53 OPC_ORI	149
10.11.1.54 OPC_SB	150
10.11.1.55 OPC_SC	150
10.11.1.56 OPC_SH	150
10.11.1.57 OPC_SLTI	150
10.11.1.58 OPC_SLTIU	150
10.11.1.59 OPC_SW	150
10.11.1.60 OPC_SWCL	150
10.11.1.61 OPC_SWL	150
10.11.1.62 OPC_SWR	151
10.11.1.63 OPC_XORI	151
10.11.1.64 RI_BGEZ	151
10.11.1.65 RI_BGEZAL	151
10.11.1.66 RI_BLTZ	151
10.11.1.67 RI_BLTZAL	151
10.11.1.68 RI_TEQI	151
10.11.1.69 RI_TGEI	151
10.11.1.70 RI_TGEIU	152

10.11.1.71 RI_TLTI	152
10.11.1.72 RI_TLIU	152
10.11.1.73 RI_TNEI	152
10.11.1.74 RLIKE_CLO	152
10.11.1.75 RLIKE_CLZ	152
10.11.1.76 RLIKE_MADD	152
10.11.1.77 RLIKE_MADDU	152
10.11.1.78 RLIKE_MSUB	153
10.11.1.79 RLIKE_MSUBU	153
10.11.1.80 RLIKE_MUL	153
10.11.2 Enumeration Type Documentation	153
10.11.2.1 TYPE	153
10.11.3 Function Documentation	153
10.11.3.1 resolv_type()	153
10.12 src/instruction_impl.cpp File Reference	154
10.13 src/instruction_impl.h File Reference	154
10.13.1 Macro Definition Documentation	156
10.13.1.1 BRANCH_IF	156
10.13.1.2 BRANCH_IF_SAVE	156
10.13.1.3 ComDef	157
10.13.1.4 ComImplDef	157
10.13.1.5 DEFAULT_INIT	157
10.13.1.6 OP_AMONG_REGS	158
10.13.1.7 OP_AMONG_REGS_OVERFLOW	158
10.13.1.8 SHIFT_IMM	158
10.13.1.9 SHIFT_REAL	159
10.13.1.10 SimDef	159
10.13.1.11 SimImplDef	159
10.13.1.12 TRAP_R	160
10.13.1.13 TRAP_RI	160
10.14 src/main.cpp File Reference	160
10.14.1 Function Documentation	160
10.14.1.1 main()	160
10.15 src/mainwindow.cpp File Reference	161
10.16 src/mainwindow.h File Reference	161
10.16.1 Macro Definition Documentation	162
10.16.1.1 CASE	162
10.16.1.2 FRAME_SIZE	162
10.16.1.3 HANDLE	162
10.16.1.4 IJCASE	162
10.16.1.5 KiB	162
10.16.1.6 MiB	163

10.16.1.7 RCASE	163
10.16.1.8 RICASE	163
10.16.1.9 RLCASE	163
10.16.2 Enumeration Type Documentation	163
10.16.2.1 MemoryType	163
10.17 src/mainwindow.ipp File Reference	164
10.17.1 Macro Definition Documentation	164
10.17.1.1 MAINWINDOW_IPP	164
10.18 src/stack.cpp File Reference	164
10.18.1 Function Documentation	164
10.18.1.1 nextPowerOfTwo()	164
10.19 src/stack.h File Reference	164
10.19.1 Macro Definition Documentation	165
10.19.1.1 DEFAULT_SIZE	165
10.19.1.2 STACK_HIGH	165
10.19.2 Function Documentation	165
10.19.2.1 nextPowerOfTwo()	165
10.20 src/syscall.h File Reference	166
10.20.1 Macro Definition Documentation	166
10.20.1.1 SYSCALL_CLOSE	166
10.20.1.2 SYSCALL_EXIT	166
10.20.1.3 SYSCALL_EXIT2	166
10.20.1.4 SYSCALL_FAST_COPY	166
10.20.1.5 SYSCALL_MMAP	167
10.20.1.6 SYSCALL_MUNMAP	167
10.20.1.7 SYSCALL_OPEN	167
10.20.1.8 SYSCALL_PRINT_CHAR	167
10.20.1.9 SYSCALL_PRINT_INT	167
10.20.1.10 SYSCALL_PRINT_STRING	167
10.20.1.11 SYSCALL_READ	167
10.20.1.12 SYSCALL_READ_CHAR	167
10.20.1.13 SYSCALL_READ_INT	168
10.20.1.14 SYSCALL_READ_STRING	168
10.20.1.15 SYSCALL_UI_OPEN_FILE	168
10.20.1.16 SYSCALL_WRITE	168

Chapter 1

MIPS Simulator



Mtoy

MIPS Simulator

A toy GUI-aided simulator for MIPS 2000 assembly language.

1.1 Introduction

This is a toy simulator of MIPS 2000 assembly environment for CSC3050 assignment 2. An assembler is integrated into the simulator. The simulator will first invoke the assembler to transform the assembly language into binary code. After that, all binary code will be decoded and transform into predefined structs in C++.

When simulation starts, the simulator will maintain the status in the registers and memory (stack/static/heap). All IO operations will be handled by GUI events.

1.2 Language Support

MIPS 2000 language without floating point instructions, coprocessor instructions and pseudo instructions. The data part supports:

- word (array)
- halfword (array)
- byte (array)
- space (array)
- ascii
- asciiz

1.3 Syscall List

Name	Code
PRINT_INT	1
PRINT_STRING	4
READ_INT	5
READ_STRING	8
MMAP	9
EXIT	10
OPEN	13
READ	14
WRITE	15
CLOSE	16
EXIT2	17
FAST_COPY	10000
FILE_OPEN_DIALOG	10001
MUNMAP	10002

1.4 Build

1.4.1 Build Environment

- Linux (64bit, GNU environment, Kernel > 3.0)
- GNU Toolchain (GCC and binary tools, Clang >= 6/GCC >= 7, GCC 9 is recommended)
- CMake (3.5 and above)
- Qt5

1.4.2 Preparation

On Ubuntu 16.04, you can install the toolchains with the following code:

```
sudo apt update
sudo apt install clang-6.0 libomp-dev libomp5
sudo apt-get update
sudo apt-get -y install clang-6.0 clang-6.0 libomp-dev libomp5 qt5-default qtbases5-dev
```

1.4.3 Compilation

On Ubuntu 16.04,

```
mkdir build && cd build  
env CC=clang-6.0 CXX=clang-6.0 cmake .. -DCMAKE_BUILD_TYPE=Release  
make -j $(nproc)
```

1.4.4 Special Notice

Older GCC on Ubuntu has a bug on `thread_local` linkage, hence it is recommended to use `llvm` toolchain on old distribution.

Chapter 2

The Design Brochure

2.1 The Procedure of Simulation

After all translation is done, the process goes like the following:

- Run the code at the current PC
 - Arithmetic Operations are done directly
 - Load/Store Operations are moved to corresponding memory handlers
 - Syscall Operations are moved to corresponding functions
- Update GUI if needed
- If exceptions/faults/exit occurs
- Advance PC if no special jumps happened

2.2 The design of MMAP/MUNMAP Simulation

We use raw `mmap` syscall with `MMAP_32BIT` flags to directly handle the syscall. To maintain the GUI part, we use a tagged statistic red-black tree from GNU's `PBDS` library to record the order of the memory begin positions. Hence, the size and memory blocks can be visualized in a list widget of GUI

2.3 SigFault Capturing

We directly use the signal capture funtions of low-level `c` library to handle the faults on memory. However, if the target operations is too destructive, the simulator will quit directly without popping a dialog.

2.4 Code Deduplication

We use a lot of macros to generate the code for us. In fact, we have designed a simple DSL to help us declare and implement an instruction.

2.5 Store the Instructions

To store the compiled instructions, we use the polymorphic classes. The binary code is "compiled" into unique pointer to the implementation classes. Hence, we do not need to do a lot of branches and dispatches in the running time.

Chapter 3

Todo List

Member `JImpl::exec` () override

: CHECK WHETHER THIS IS REQUIRED

Chapter 4

Namespace Index

4.1 Namespace List

Here is a list of all namespaces with brief descriptions:

_SIM	17
Ui	19

Chapter 5

Hierarchical Index

5.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Heap	46
_SIM::InstrDeleter	48
InstructionImpl	49
ADDImpl	21
ADDImpl	22
ADDIImpl	23
ADDUImpl	24
ANDImpl	25
ANDImpl	26
BEQImpl	27
BGEZALImpl	28
BGEZALLImpl	29
BGEZImpl	30
BGEZLImpl	31
BGTZImpl	32
BLEZImpl	33
BLTZALImpl	34
BLTZALLImpl	35
BLTZImpl	36
BLTZLImpl	37
BNEImpl	38
BREAKImpl	39
CLOImpl	40
CLZImpl	41
DIVImpl	42
DIVUImpl	43
JImpl	54
JALImpl	52
JRImpl	55
JALRImpl	53
LBImpl	57
LBUImpl	58
LHImpl	59
LHUImpl	60
LUIImpl	62

LWImpl	63
LLImpl	61
LWLImpl	64
LWRImpl	65
MADDImpl	66
MADDUImpl	67
MFHImpl	82
MFLOImpl	83
MSUBImpl	84
MSUBUImpl	85
MTHImpl	86
MTLOImpl	87
MULImpl	88
MULTImpl	89
MULTUImpl	90
NOPImpl	91
NORImpl	92
ORImpl	93
ORImpl	94
SBImpl	95
SHImpl	97
SLLImpl	98
SLLVImpl	99
SLTImpl	100
SLTImpl	101
SLTIUImpl	102
SLTUImpl	103
SRAImpl	104
SRAVImpl	105
SRLImpl	106
SRLVImpl	107
SUBImpl	112
SUBUImpl	113
SWCLImpl	114
SWImpl	115
SCImpl	96
SWLImpl	116
SWRImpl	117
SYSCALLImpl	118
TEQImpl	119
TEQImpl	120
TGEImpl	121
TGEImpl	122
TGEIImpl	123
TGEUImpl	124
TLTImpl	125
TLTImpl	126
TLTIUImpl	127
TLTUImpl	128
TNEImpl	129
TNEImpl	130
XORImpl	131
XORImpl	132
QMainWindow	
MainWindow	68
QObject	
Executor	44
Stack	108

Chapter 6

Class Index

6.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

ADDImpl	21
ADDImpl	22
ADDUIImpl	23
ADDUIImpl	24
ANDImpl	25
ANDImpl	26
BEQImpl	27
BGEZALImpl	28
BGEZALLImpl	29
BGEZImpl	30
BGEZLImpl	31
BGTZImpl	32
BLEZImpl	33
BLTZALImpl	34
BLTZALLImpl	35
BLTZImpl	36
BLTZLImpl	37
BNEImpl	38
BREAKImpl	39
CLOImpl	40
CLZImpl	41
DIVImpl	42
DIVUIImpl	43
Executor	44
Heap	46
_SIM::InstrDeleter	48
InstructionImpl	49
JALImpl	52
JALRImpl	53
JImpl	54
JRImpl	55
LBImp	57
LBUImpl	58
LHImpl	59
LHUImpl	60

LLImpl	61
LUIImpl	62
LWImpl	63
LWLImpl	64
LWRImpl	65
MADDImpl	66
MADDUImpl	67
MainWindow	68
MFHIImpl	82
MFLOImpl	83
MSUBImpl	84
MSUBUImpl	85
MTHIImpl	86
MTLOImpl	87
MULImpl	88
MULTImpl	89
MULTUImpl	90
NOPImpl	91
NORImpl	92
ORIImpl	93
ORImpl	94
SBIImpl	95
SCTImpl	96
SHImpl	97
SLLImpl	98
SLLVImpl	99
SLTIImpl	100
SLTImpl	101
SLTIUImpl	102
SLTUImpl	103
SRAImpl	104
SRAVImpl	105
SRLImpl	106
SRLVImpl	107
Stack	108
SUBImpl	112
SUBUImpl	113
SWCLImpl	114
SWImpl	115
SWLImpl	116
SWRImpl	117
SYSCALLImpl	118
TEQIImpl	119
TEQImpl	120
TGEIImpl	121
TGEImpl	122
TGEIUImpl	123
TGEUImpl	124
TLTIImpl	125
TLTImpl	126
TLTIUImpl	127
TLTUImpl	128
TNEIImpl	129
TNEImpl	130
XORIImpl	131
XORImpl	132

Chapter 7

File Index

7.1 File List

Here is a list of all files with brief descriptions:

src/executor.cpp	135
src/executor.h	135
src/fs.h	135
src/global.cpp	136
src/global.h	136
src/heap.cpp	138
src/heap.h	139
src/instruction.cpp	140
src/instruction.h	141
src/instruction_impl.cpp	154
src/instruction_impl.h	154
src/main.cpp	160
src/mainwindow.cpp	161
src/mainwindow.h	161
src/mainwindow.ipp	164
src/stack.cpp	164
src/stack.h	164
src/syscall.h	166

Chapter 8

Namespace Documentation

8.1 `_SIM` Namespace Reference

Classes

- struct `InstrDeleter`

Typedefs

- using `InstrPtr` = `std::unique_ptr< InstructionImpl, InstrDeleter >`

Functions

- `template<typename T, typename ... Args>`
`InstrPtr make_unique (Args &&...args)`

8.1.1 Detailed Description

This namespace contains some workarounds to provide back support

8.1.2 Typedef Documentation

8.1.2.1 `InstrPtr`

```
using _SIM::InstrPtr = typedef std::unique_ptr<InstructionImpl, InstrDeleter>
```

Alias of the `unique_ptr` of instruction implementations

8.1.3 Function Documentation

8.1.3.1 `make_unique()`

```
template<typename T , typename ... Args>  
InstrPtr _SIM::make_unique (   
    Args &&... args )
```

C++ 11 do not have `make_unique`. This functions works as an substitution.

Template Parameters

<i>T</i>	data type
<i>Args</i>	initialization args type

Parameters

<i>args</i>	initialization args
-------------	---------------------

Returns

the unique ptr

8.2 Ui Namespace Reference

8.2.1 Detailed Description

The Qt specified namespace for UI components

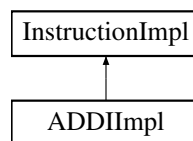
Chapter 9

Class Documentation

9.1 ADDImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for ADDImpl:



Public Member Functions

- `ADDImpl` (Instruction *instr*)
- void `exec` () override

Additional Inherited Members

9.1.1 Constructor & Destructor Documentation

9.1.1.1 ADDImpl()

```
ADDImpl::ADDImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.1.2 Member Function Documentation

9.1.2.1 exec()

```
void ADDImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

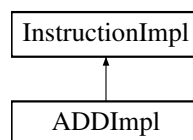
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.2 ADDImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for ADDImpl:



Public Member Functions

- [ADDImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.2.1 Constructor & Destructor Documentation

9.2.1.1 ADDImpl()

```
ADDImpl::ADDImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.2.2 Member Function Documentation

9.2.2.1 exec()

```
void ADDIUImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

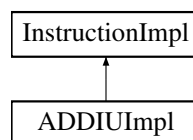
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.3 ADDIUImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for ADDIUImpl:



Public Member Functions

- [ADDIUImpl](#) (Instruction *instr*)
- void [exec](#) () override

Additional Inherited Members

9.3.1 Constructor & Destructor Documentation

9.3.1.1 ADDIUImpl()

```
ADDIUImpl::ADDIUImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.3.2 Member Function Documentation

9.3.2.1 exec()

```
void ADDUImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

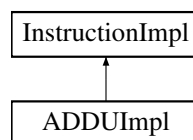
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.4 ADDUImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for ADDUImpl:



Public Member Functions

- [ADDUImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.4.1 Constructor & Destructor Documentation

9.4.1.1 ADDUImpl()

```
ADDUImpl::ADDUImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.4.2 Member Function Documentation

9.4.2.1 exec()

```
void ADDUImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

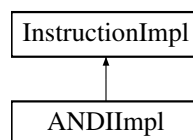
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.5 ANDImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for ANDImpl:



Public Member Functions

- [ANDImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.5.1 Constructor & Destructor Documentation

9.5.1.1 ANDImpl()

```
ANDImpl::ANDImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.5.2 Member Function Documentation

9.5.2.1 exec()

```
void ANDImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

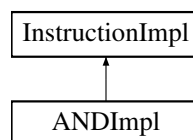
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.6 ANDImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for ANDImpl:



Public Member Functions

- [ANDImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.6.1 Constructor & Destructor Documentation

9.6.1.1 ANDImpl()

```
ANDImpl::ANDImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.6.2 Member Function Documentation

9.6.2.1 exec()

```
void ANDImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

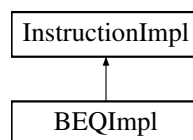
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.7 BEQImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for BEQImpl:



Public Member Functions

- [BEQImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.7.1 Constructor & Destructor Documentation

9.7.1.1 BEQImpl()

```
BEQImpl::BEQImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.7.2 Member Function Documentation

9.7.2.1 exec()

```
void BEQImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

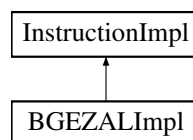
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.8 BGEZALImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for BGEZALImpl:



Public Member Functions

- [BGEZALImpl](#) (Instruction *instr*)
- void [exec](#) () override

Additional Inherited Members

9.8.1 Constructor & Destructor Documentation

9.8.1.1 BGEZALImpl()

```
BGEZALImpl::BGEZALImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.8.2 Member Function Documentation

9.8.2.1 exec()

```
void BGEZALLImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

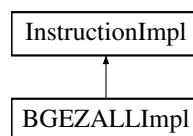
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.9 BGEZALLImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for BGEZALLImpl:



Public Member Functions

- [BGEZALLImpl](#) (Instruction *instr*)
- void [exec](#) () override

Additional Inherited Members

9.9.1 Constructor & Destructor Documentation

9.9.1.1 BGEZALLImpl()

```
BGEZALLImpl::BGEZALLImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.9.2 Member Function Documentation

9.9.2.1 exec()

```
void BGEZALLImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

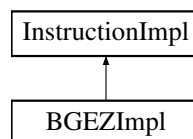
The documentation for this struct was generated from the following file:

- [src/instruction_impl.h](#)

9.10 BGEZImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for BGEZImpl:



Public Member Functions

- [BGEZImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.10.1 Constructor & Destructor Documentation

9.10.1.1 BGEZImpl()

```
BGEZImpl::BGEZImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.10.2 Member Function Documentation

9.10.2.1 exec()

```
void BGEZLImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

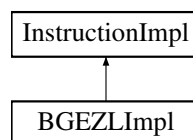
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.11 BGEZLImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for BGEZLImpl:



Public Member Functions

- [BGEZLImpl](#) (Instruction *instr*)
- void [exec](#) () override

Additional Inherited Members

9.11.1 Constructor & Destructor Documentation

9.11.1.1 BGEZLImpl()

```
BGEZLImpl::BGEZLImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.11.2 Member Function Documentation

9.11.2.1 `exec()`

```
void BGEZLImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

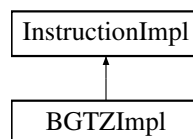
The documentation for this struct was generated from the following file:

- [src/instruction_impl.h](#)

9.12 BGTZImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for BGTZImpl:



Public Member Functions

- [BGTZImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.12.1 Constructor & Destructor Documentation

9.12.1.1 BGTZImpl()

```
BGTZImpl::BGTZImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.12.2 Member Function Documentation

9.12.2.1 exec()

```
void BGTZImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

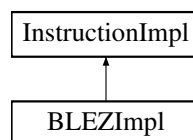
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.13 BLEZImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for BLEZImpl:



Public Member Functions

- [BLEZImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.13.1 Constructor & Destructor Documentation

9.13.1.1 BLEZImpl()

```
BLEZImpl::BLEZImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.13.2 Member Function Documentation

9.13.2.1 exec()

```
void BLEZImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

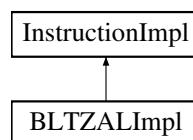
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.14 BLTZALImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for BLTZALImpl:



Public Member Functions

- [BLTZALImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.14.1 Constructor & Destructor Documentation

9.14.1.1 BLTZALImpl()

```
BLTZALImpl::BLTZALImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.14.2 Member Function Documentation

9.14.2.1 exec()

```
void BLTZALLImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

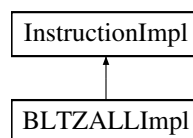
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.15 BLTZALLImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for BLTZALLImpl:



Public Member Functions

- [BLTZALLImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.15.1 Constructor & Destructor Documentation

9.15.1.1 BLTZALLImpl()

```
BLTZALLImpl::BLTZALLImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.15.2 Member Function Documentation

9.15.2.1 exec()

```
void BLTZALLImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

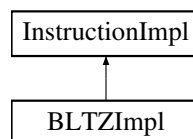
The documentation for this struct was generated from the following file:

- [src/instruction_impl.h](#)

9.16 BLTZImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for BLTZImpl:



Public Member Functions

- [BLTZImpl](#) (Instruction *instr*)
- void [exec](#) () override

Additional Inherited Members

9.16.1 Constructor & Destructor Documentation

9.16.1.1 BLTZImpl()

```
BLTZImpl::BLTZImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.16.2 Member Function Documentation

9.16.2.1 exec()

```
void BLTZLImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

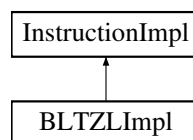
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.17 BLTZLImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for BLTZLImpl:



Public Member Functions

- [BLTZLImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.17.1 Constructor & Destructor Documentation

9.17.1.1 BLTZLImpl()

```
BLTZLImpl::BLTZLImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.17.2 Member Function Documentation

9.17.2.1 `exec()`

```
void BLTZLImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

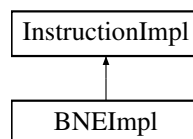
The documentation for this struct was generated from the following file:

- [src/instruction_impl.h](#)

9.18 BNEImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for BNEImpl:



Public Member Functions

- [BNEImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.18.1 Constructor & Destructor Documentation

9.18.1.1 `BNEImpl()`

```
BNEImpl::BNEImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.18.2 Member Function Documentation

9.18.2.1 exec()

```
void BNEImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

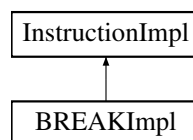
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.19 BREAKImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for BREAKImpl:



Public Member Functions

- [BREAKImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.19.1 Constructor & Destructor Documentation

9.19.1.1 BREAKImpl()

```
BREAKImpl::BREAKImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.19.2 Member Function Documentation

9.19.2.1 exec()

```
void BREAKImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

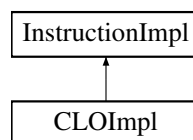
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.20 CLOImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for CLOImpl:



Public Member Functions

- [CLOImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.20.1 Constructor & Destructor Documentation

9.20.1.1 CLOImpl()

```
CLOImpl::CLOImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.20.2 Member Function Documentation

9.20.2.1 exec()

```
void CLZImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

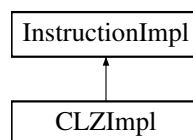
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.21 CLZImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for CLZImpl:



Public Member Functions

- [CLZImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.21.1 Constructor & Destructor Documentation

9.21.1.1 CLZImpl()

```
CLZImpl::CLZImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.21.2 Member Function Documentation

9.21.2.1 exec()

```
void CLZImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

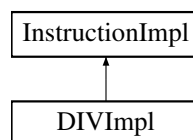
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.22 DIVImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for DIVImpl:



Public Member Functions

- [DIVImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.22.1 Constructor & Destructor Documentation

9.22.1.1 DIVImpl()

```
DIVImpl::DIVImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.22.2 Member Function Documentation

9.22.2.1 exec()

```
void DIVUImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

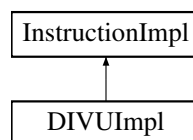
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.23 DIVUImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for DIVUImpl:



Public Member Functions

- [DIVUImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.23.1 Constructor & Destructor Documentation

9.23.1.1 DIVUImpl()

```
DIVUImpl::DIVUImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.23.2 Member Function Documentation

9.23.2.1 exec()

```
void DIVUImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

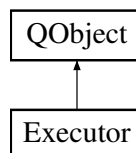
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.24 Executor Class Reference

```
#include <executor.h>
```

Inheritance diagram for Executor:



Public Slots

- void [next](#) ()
- void [exit](#) (int code=0)

Signals

- void [finished](#) ()

Public Attributes

- `std::vector< _SIM::InstrPtr > impls`
a set of pointers to the translated result of the machine code.

Static Public Attributes

- static [MainWindow](#) * [mainW](#) = nullptr
a pointer back to the [MainWindow](#), provides APIs on the UI components.

9.24.1 Detailed Description

[Executor](#) is a helper class for us to store compiled instructions and enable slots for automatic execution.

9.24.2 Member Function Documentation

9.24.2.1 exit

```
void Executor::exit (
    int code = 0 ) [slot]
```

Exit the execution with the given code. (Information is shown) It will also emit a finish signal on the closing of the information dialog.

Parameters

<i>code</i>	return code
-------------	-------------

9.24.2.2 finished

```
void Executor::finished ( ) [signal]
```

This signal will be emitted when the execution is finished.

9.24.2.3 next

```
void Executor::next ( ) [slot]
```

When this slot is invoked, it will execute the instruction at the current PC position. When PC touches the bottom line, it will emit a finish signal.

Attention

`std::runtime_error` and `SIGSEGV` will be caught within the range.

9.24.3 Member Data Documentation

9.24.3.1 impls

```
std::vector<_SIM::InstrPtr> Executor::impls
```

a set of pointers to the translated result of the machine code.

9.24.3.2 mainW

```
MainWindow * Executor::mainW = nullptr [static]
```

a pointer back to the [MainWindow](#), provides APIs on the UI components.

The documentation for this class was generated from the following files:

- [src/executor.h](#)
- [src/executor.cpp](#)
- [src/mainwindow.cpp](#)

9.25 Heap Class Reference

```
#include <heap.h>
```

Public Member Functions

- void [clear](#) ()
- [~Heap](#) ()
- uint32_t [alloc](#) (size_t n)
- void [dealloc](#) (uint32_t addr)
- size_t [order](#) (uint32_t addr)

Public Attributes

- size_t [size](#) = 0
The size of current heap.

Private Attributes

- [TreeSet](#)< uint32_t, size_t > [mapping](#)
Used to store the block size of each allocation.

9.25.1 Detailed Description

[Heap](#) help us to record and manage heap storage

9.25.2 Constructor & Destructor Documentation

9.25.2.1 ~Heap()

```
Heap::~Heap ( )
```

Deconstruct the heap, will do the same thing as [clear\(\)](#);

9.25.3 Member Function Documentation

9.25.3.1 alloc()

```
uint32_t Heap::alloc (
    size_t n )
```

Alloc a new memory block.

Parameters

<i>n</i>	memory block size
----------	-------------------

Returns

the address of the newly allocated memory block

9.25.3.2 clear()

```
void Heap::clear ( )
```

Clear all heap content

9.25.3.3 dealloc()

```
void Heap::dealloc (
    uint32_t addr )
```

Dealloc a memory block

Parameters

<i>addr</i>	the start address of the memory block.
-------------	--

9.25.3.4 order()

```
size_t Heap::order (
    uint32_t addr )
```

Check the order of an address. This is used in UI display to check which block to remove from the list.

Parameters

<i>addr</i>	the address to check
-------------	----------------------

Returns

the order of the block

9.25.4 Member Data Documentation**9.25.4.1 mapping**

```
TreeSet<uint32_t, size_t> Heap::mapping [private]
```

Used to store the block size of each allocation.

9.25.4.2 size

```
size_t Heap::size = 0
```

The size of current heap.

The documentation for this class was generated from the following files:

- [src/heap.h](#)
- [src/heap.cpp](#)

9.26 _SIM::InstrDeleter Struct Reference

```
#include <global.h>
```

Public Member Functions

- void [operator\(\)](#) ([InstructionImpl](#) *t)

9.26.1 Detailed Description

The deleter for the unique_ptr of instruction implementations

9.26.2 Member Function Documentation

9.26.2.1 operator()

```
void _SIM::InstrDeleter::operator() (
    InstructionImpl * t ) [inline]
```

The documentation for this struct was generated from the following file:

- [src/global.h](#)

9.27 InstructionImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for InstructionImpl:



Public Member Functions

- virtual void `exec()`=0
- `InstructionImpl` (Instruction `instr`)

Public Attributes

- Instruction `instr`
the instruction value.

Static Public Attributes

- static `MainWindow * mainW` = nullptr

9.27.1 Detailed Description

The base class of the instruction implementations. All subclasses must implement the `exec()` function, and this function decides the behavior of each construction.

9.27.2 Constructor & Destructor Documentation

9.27.2.1 `InstructionImpl()`

```
InstructionImpl::InstructionImpl (  
    Instruction instr ) [explicit]
```

Construct the base class: `InstructionImpl`.

Attention

This function will set the instruction and should be invoked by all subclasses in some way.

Parameters

<code>instr</code>	the instruction value
--------------------	-----------------------

9.27.3 Member Function Documentation

9.27.3.1 exec()

```
virtual void InstructionImpl::exec ( ) [pure virtual]
```

The purely virtual function `exec ()` is to be overwritten in the subclasses.

Implemented in [LLImpl](#), [SCImpl](#), [LWRImpl](#), [LWLImpl](#), [SWRImpl](#), [SWLImpl](#), [TLTUIImpl](#), [TLTImpl](#), [TGEUIImpl](#), [TGEImpl](#), [TNEImpl](#), [TEQImpl](#), [MSUBUIImpl](#), [MSUBImpl](#), [MADDUIImpl](#), [MADDImpl](#), [MULImpl](#), [CLZImpl](#), [CLOImpl](#), [BGEZALLImpl](#), [BLTZALLImpl](#), [BGEZALImpl](#), [BLTZALImpl](#), [TNEIImpl](#), [TEQIImpl](#), [TLTIUIImpl](#), [TLTIImpl](#), [TGEIUIImpl](#), [TGEIImpl](#), [BGEZLImpl](#), [BLTZLImpl](#), [XORImpl](#), [SYSCALLImpl](#), [SUBUIImpl](#), [SUBImpl](#), [SRLVImpl](#), [SRLImpl](#), [SRAVImpl](#), [SRAImpl](#), [SLTUIImpl](#), [SLTImpl](#), [SLLVImpl](#), [SLLImpl](#), [ORImpl](#), [NORImpl](#), [MULTUIImpl](#), [MULTImpl](#), [MTLOImpl](#), [MTHIImpl](#), [MFLOImpl](#), [MFHIImpl](#), [JALRImpl](#), [JRImpl](#), [DIVUIImpl](#), [DIVImpl](#), [BREAKImpl](#), [ANDImpl](#), [ADDUIImpl](#), [ADDImpl](#), [XORIImpl](#), [SWCLImpl](#), [SWImpl](#), [SHImpl](#), [SLTIUIImpl](#), [SLTIImpl](#), [SBIImpl](#), [ORIImpl](#), [LWImpl](#), [LUIImpl](#), [LHUImpl](#), [LHImpl](#), [LBUImpl](#), [LBImpl](#), [BNEImpl](#), [BLTZImpl](#), [BLEZImpl](#), [BGTZImpl](#), [BGEZImpl](#), [BEQImpl](#), [ANDIImpl](#), [ADDIUIImpl](#), [ADDIImpl](#), [JALIImpl](#), [JImpl](#), and [NOPIImpl](#).

9.27.4 Member Data Documentation

9.27.4.1 instr

```
Instruction InstructionImpl::instr
```

the instruction value.

9.27.4.2 mainW

```
MainWindow * InstructionImpl::mainW = nullptr [static]
```

Static pointer back to the main windows. This pointer allow us to interact with the ui components.

Attention

This pointer should never be modified in any situation except for the setting happens during the initialization [MainWindow](#)

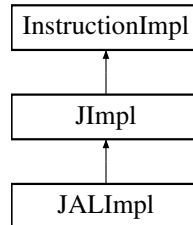
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.28 JALImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for JALImpl:



Public Member Functions

- [JALImpl](#) (Instruction *instr*)
- void [exec](#) () override

Additional Inherited Members

9.28.1 Constructor & Destructor Documentation

9.28.1.1 JALImpl()

```
JALImpl::JALImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.28.2 Member Function Documentation

9.28.2.1 exec()

```
void JALImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Reimplemented from [JImpl](#).

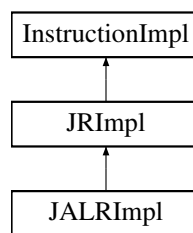
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.29 JALRImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for JALRImpl:



Public Member Functions

- [JALRImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.29.1 Constructor & Destructor Documentation

9.29.1.1 JALRImpl()

```
JALRImpl::JALRImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.29.2 Member Function Documentation

9.29.2.1 exec()

```
void JALRImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Reimplemented from [JRImpl](#).

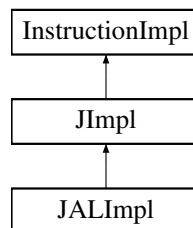
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.30 JImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for JImpl:



Public Member Functions

- [JImpl](#) (Instruction *instr*)
- void [exec](#) () override

Additional Inherited Members

9.30.1 Constructor & Destructor Documentation

9.30.1.1 JImpl()

```
JImpl::JImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.30.2 Member Function Documentation

9.30.2.1 exec()

```
void JRImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Todo : CHECK WHETHER THIS IS REQUIRED

Implements [InstructionImpl](#).

Reimplemented in [JALRImpl](#).

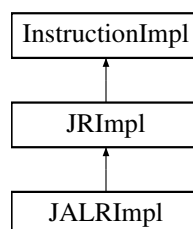
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.31 JRImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for JRImpl:



Public Member Functions

- [JRImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.31.1 Constructor & Destructor Documentation

9.31.1.1 JRImpl()

```
JRImpl::JRImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.31.2 Member Function Documentation

9.31.2.1 exec()

```
void JRImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

Reimplemented in [JALRImpl](#).

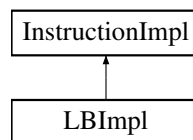
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.32 LBImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for LBImpl:



Public Member Functions

- [LBImpl](#) (Instruction *instr*)
- void [exec](#) () override

Additional Inherited Members

9.32.1 Constructor & Destructor Documentation

9.32.1.1 LBImpl()

```
LBImpl::LBImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.32.2 Member Function Documentation

9.32.2.1 exec()

```
void LBImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

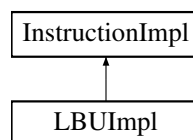
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.33 LBImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for LBImpl:



Public Member Functions

- [LBImpl](#) (Instruction *instr*)
- void [exec](#) () override

Additional Inherited Members

9.33.1 Constructor & Destructor Documentation

9.33.1.1 LBImpl()

```
LBImpl::LBImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.33.2 Member Function Documentation

9.33.2.1 exec()

```
void LBUIImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

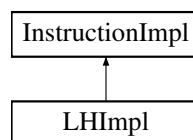
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.34 LHImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for LHImpl:



Public Member Functions

- [LHImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.34.1 Constructor & Destructor Documentation

9.34.1.1 LHImpl()

```
LHImpl::LHImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.34.2 Member Function Documentation

9.34.2.1 exec()

```
void LHImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

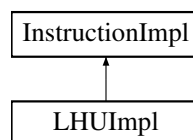
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.35 LHUImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for LHUImpl:



Public Member Functions

- [LHUImpl](#) (Instruction *instr*)
- void [exec](#) () override

Additional Inherited Members

9.35.1 Constructor & Destructor Documentation

9.35.1.1 LHUImpl()

```
LHUImpl::LHUImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.35.2 Member Function Documentation

9.35.2.1 exec()

```
void LHUImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

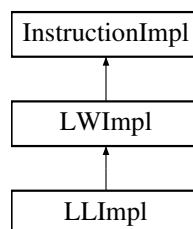
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.36 LLImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for LLImpl:



Public Member Functions

- [LLImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.36.1 Constructor & Destructor Documentation

9.36.1.1 LLImpl()

```
LLImpl::LLImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.36.2 Member Function Documentation

9.36.2.1 exec()

```
void LLImp::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Attention

we have no difference between atomic ones and unatomic ones.

Reimplemented from [LWImpl](#).

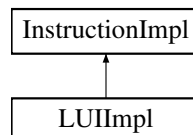
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.37 LUIImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for LUIImpl:



Public Member Functions

- [LUIImpl](#) (Instruction *instr*)
- void [exec](#) () override

Additional Inherited Members

9.37.1 Constructor & Destructor Documentation

9.37.1.1 LUIImpl()

```
LUIImpl::LUIImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.37.2 Member Function Documentation

9.37.2.1 exec()

```
void LWImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

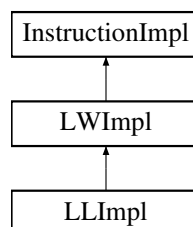
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.38 LWImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for LWImpl:



Public Member Functions

- [LWImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.38.1 Constructor & Destructor Documentation

9.38.1.1 LWImpl()

```
LWImpl::LWImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.38.2 Member Function Documentation

9.38.2.1 exec()

```
void LWImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

Reimplemented in [LLImpl](#).

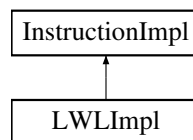
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.39 LWLImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for LWLImpl:



Public Member Functions

- [LWLImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.39.1 Constructor & Destructor Documentation

9.39.1.1 LWLImpl()

```
LWLImpl::LWLImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.39.2 Member Function Documentation

9.39.2.1 exec()

```
void LWRImpI::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

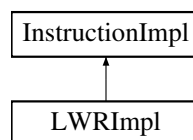
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.40 LWRImpI Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for LWRImpI:



Public Member Functions

- [LWRImpI](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.40.1 Constructor & Destructor Documentation

9.40.1.1 LWRImpI()

```
LWRImpI::LWRImpI (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.40.2 Member Function Documentation

9.40.2.1 exec()

```
void LWRImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

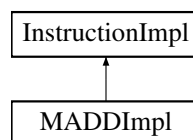
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.41 MADDImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for MADDImpl:



Public Member Functions

- [MADDImpl](#) (Instruction *instr*)
- void [exec](#) () override

Additional Inherited Members

9.41.1 Constructor & Destructor Documentation

9.41.1.1 MADDImpl()

```
MADDImpl::MADDImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.41.2 Member Function Documentation

9.41.2.1 exec()

```
void MADDUImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

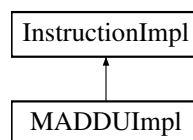
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.42 MADDUImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for MADDUImpl:



Public Member Functions

- [MADDUImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.42.1 Constructor & Destructor Documentation

9.42.1.1 MADDUImpl()

```
MADDUImpl::MADDUImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.42.2 Member Function Documentation

9.42.2.1 exec()

```
void MADDUImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

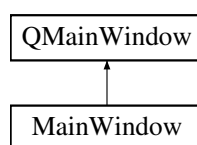
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.43 MainWindow Class Reference

```
#include <mainwindow.h>
```

Inheritance diagram for MainWindow:



Public Member Functions

- [MainWindow](#) (QWidget *parent=nullptr)
- [~MainWindow](#) () override
- void [showWarning](#) (QString str)
- void [updateRegValue](#) (int no, uint32_t value, const QBrush &brush=QBrush("red"), bool init=false)
- void [updateProgramCounter](#) (size_t value)
- uint32_t [allocHeap](#) (size_t size)
- void [deallocHeap](#) (size_t addr)
- template<class T >
 T [fetchHeap](#) (uint32_t addr)
- template<class T >
 void [editHeap](#) (uint32_t addr, T value)

- template<class T >
void [edit](#) (uint32_t addr, T value)
- void [increaseStack](#) (size_t n)
- void [decreaseStack](#) (size_t n)
- template<class T >
T & [fetchStack](#) (uint32_t addr)
- template<class T >
void [editStack](#) (uint32_t addr, T value)
- [MemoryType](#) [memoryType](#) (uint32_t addr)
- void [updateStack](#) (uint32_t addr, size_t size)
- void [updateLow](#) (uint32_t value)
- void [updateHigh](#) (uint32_t value)
- void [updateAcc](#) (uint64_t value)
- void [translateAll](#) ()
- void [resetAll](#) ()
- template<class T >
T * [getRealAddr](#) (uint32_t addr)
- void [handleSyscall](#) ()

Public Attributes

- union {
 uint64_t [all](#)
 stands for the whole accumulator
 struct **LOW_HIGH** {
 uint32_t [low](#)
 stands for the low part of the accumulator
 uint32_t [high](#)
 stands for the high part of the accumulator
 } [part](#)
} [ACC](#)
- std::array< char,(1024 *[KiB](#)) > [frame](#)
- Ui::MainWindow * [ui](#)
The pointer to the Qt UI components.
- [Executor](#) * [executor](#) = nullptr
The pointer to the executor.
- bool [advanceCounter](#) = true
- QTimer [timer](#)
A timer to ignite periodical signals on execution.
- std::vector< Instruction > [instructions](#) {}
The vector for instruction storage.
- [Stack](#) [stack](#)
The stack operation simulator.
- [Heap](#) [heap](#)
The heap operation simulator.
- uint32_t [REGS](#) [32] = {}
The array for all registers.
- size_t [PC](#) = 0
The program counter.

Private Slots

- void [on_aboutButton_clicked](#) ()
- void [on_openButton_clicked](#) ()
- void [on_translateButton_clicked](#) ()
- void [on_executeButton_clicked](#) ()
- void [on_stepButton_clicked](#) ()
- void [on_resetButton_clicked](#) ()
- void [on_stopButton_clicked](#) ()
- void [on_pushButton_clicked](#) ()

9.43.1 Detailed Description

[MainWindow](#) of the GUI program. It also contains the following component:

- a heap to simulate heap allocations
- a stack to simulate stack operations
- an executor to store and run simulation steps

Attention

we keep all fields in public to reduce the complexity of development

9.43.2 Constructor & Destructor Documentation

9.43.2.1 MainWindow()

```
MainWindow::MainWindow (
    QWidget * parent = nullptr ) [explicit]
```

construct the [MainWindow](#)

Parameters

<i>parent</i>	the parent widget, set as nullptr by default.
---------------	---

On initialization, it will:

- adjust the scale factor of the tables and lists.
- disable execution related buttons
- set initial values of the registers and other simulation storage
- capture all SIGSEGV signals and throw errors on caught

9.43.2.2 ~MainWindow()

```
MainWindow::~MainWindow ( ) [override]
```

deconstruct the MainWindows, this will:

- delete all ui components in the tree
- delete the executor
- the stack and heap space are freed automatically (since they are normal data members)

9.43.3 Member Function Documentation

9.43.3.1 allocHeap()

```
uint32_t MainWindow::allocHeap (
    size_t size )
```

Allocate some new memory on heap. This operation will update the UI components in the same time.

Attention

this will truly invoke the `mmap` syscall in your bare metal machine. For x64_64 linux target, we are using `MALLOC_32BIT` as the flag to force the system give an available address in the first 2GiB area. In the simulation, we are actually storing the size in an associate set. The allocation and deallocation operations are much slower than the real world, but as it is just a simulator with heavy UI animations, the cost is bearable.

Parameters

<i>size</i>	the required size of the allocation
-------------	-------------------------------------

Returns

the pointer to the beginning of the newly allocated memory.

9.43.3.2 deallocHeap()

```
void MainWindow::deallocHeap (
    size_t addr )
```

Deallocate some memory on heap. This operation will update the UI components in the same time.

Parameters

<i>addr</i>	the address to dealloc
-------------	------------------------

Attention

In our simulator, the following things are required:

- the pointer should point to the start position of the memory block
- the memory is managed by the heap

9.43.3.3 decreaseStack()

```
void MainWindow::decreaseStack (
    size_t n )
```

Decrease the stack size by the given amount. This will update the UI at the same time.

Parameters

<i>n</i>	the amount to decrease
----------	------------------------

9.43.3.4 edit()

```
template<class T >
void MainWindow::edit (
    uint32_t addr,
    T value )
```

9.43.3.5 editHeap()

```
template<class T >
void MainWindow::editHeap (
    uint32_t addr,
    T value )
```

Modify the target word cell on heap.

Template Parameters

<i>T</i>	T primitive word type
----------	-----------------------

Parameters

<i>addr</i>	the target address
<i>value</i>	the new value

9.43.3.6 editStack()

```
template<class T >
void MainWindow::editStack (
    uint32_t addr,
    T value )
```

Edit the stack value and update the ui display

Template Parameters

<i>T</i>	primitive type
----------	----------------

Parameters

<i>addr</i>	virtual address on stack
<i>value</i>	the value to be set

9.43.3.7 fetchHeap()

```
template<class T >
T MainWindow::fetchHeap (
    uint32_t addr )
```

Read a target word from heap.

Template Parameters

<i>T</i>	primitive word type
----------	---------------------

Parameters

<i>addr</i>	the target address
-------------	--------------------

Returns

the required word value

9.43.3.8 fetchStack()

```
template<class T >
T & MainWindow::fetchStack (
    uint32_t addr )
```

get the reference of the T value at an address on stack

Template Parameters

<i>T</i>	primitive type
----------	----------------

Parameters

<i>addr</i>	virtual address on stack
-------------	--------------------------

Returns

the reference to the target data

9.43.3.9 getRealAddr()

```
template<class T >
T * MainWindow::getRealAddr (
    uint32_t addr )
```

get the real address of the memory

Template Parameters

<i>T</i>	data type
----------	-----------

Parameters

<i>addr</i>	simulated memory
-------------	------------------

Returns

the real address

9.43.3.10 handleSyscall()

```
void MainWindow::handleSyscall ( )
```

handle syscall based on the value in the target registers

9.43.3.11 increaseStack()

```
void MainWindow::increaseStack (
    size_t n )
```

Increase the stack size by the given amount. This will update the UI at the same time.

Parameters

<i>n</i>	the amount to increase
----------	------------------------

9.43.3.12 memoryType()

```
MemoryType MainWindow::memoryType (
    uint32_t addr )
```

Check whether a address is currently within the stack range.

Parameters

<i>addr</i>	the address to check
-------------	----------------------

Returns

the checking result

9.43.3.13 on_aboutButton_clicked

```
void MainWindow::on_aboutButton_clicked ( ) [private], [slot]
```

The slot for the `aboutButton`, on click, it will pop up a about window with icon and basic information.

9.43.3.14 on_executeButton_clicked

```
void MainWindow::on_executeButton_clicked ( ) [private], [slot]
```

The slot for the `executionButton`, on click, it will start the execution. It will also:

- set the time interval set in the `delay`
- disable buttons and inputs that will change the execution behavior
- enable the `stopButton`

9.43.3.15 on_openButton_clicked

```
void MainWindow::on_openButton_clicked ( ) [private], [slot]
```

The slot for the `openButton`, on click, it will open a file chooser for the user to choose the compiled asm file.

Attention

Notice that if the file format is not correct, an error will be raised with a warning dialog.

9.43.3.16 on_pushButton_clicked

```
void MainWindow::on_pushButton_clicked ( ) [private], [slot]
```

9.43.3.17 on_resetButton_clicked

```
void MainWindow::on_resetButton_clicked ( ) [private], [slot]
```

The slot for the `stepButton`, on click, it will reset all state.

Attention

Notice that `reset` will also clear all instructions, you will need to open your file again

9.43.3.18 on_stepButton_clicked

```
void MainWindow::on_stepButton_clicked ( ) [private], [slot]
```

The slot for the `stepButton`, on click, it will do a single line execution.

9.43.3.19 on_stopButton_clicked

```
void MainWindow::on_stopButton_clicked ( ) [private], [slot]
```

The slot for the `stepButton`, on click, it will stop the execution. The PC will remain unchanged and the execution can be resumed.

9.43.3.20 on_translateButton_clicked

```
void MainWindow::on_translateButton_clicked ( ) [private], [slot]
```

The slot for the `translateButton`, on click, it will start the parallel translation. After translation, it will enable those execution related buttons. It will free the current executor and try to create a new one.

Attention

Notice that if a binary string is not recognized, the translation procedure will not stop immediately. It will store the error and give a whole summary after all lines are checked and translated.

9.43.3.21 resetAll()

```
void MainWindow::resetAll ( )
```

Clean all data

9.43.3.22 showWarning()

```
void MainWindow::showWarning (
    QString str )
```

Pop up a warning dialog.

Parameters

<i>str</i>	the warning message
------------	---------------------

9.43.3.23 translateAll()

```
void MainWindow::translateAll ( )
```

Translate all instructions

9.43.3.24 updateAcc()

```
void MainWindow::updateAcc (
    uint64_t value )
```

Update the entire accumulator

Parameters

<i>value</i>	new value
--------------	-----------

9.43.3.25 updateHigh()

```
void MainWindow::updateHigh (
    uint32_t value )
```

Update the higher bits in the accumulator

Parameters

<i>value</i>	new value
--------------	-----------

9.43.3.26 updateLow()

```
void MainWindow::updateLow (
    uint32_t value )
```

Update the lower bits in the accumulator

Parameters

<i>value</i>	new value
--------------	-----------

9.43.3.27 updateProgramCounter()

```
void MainWindow::updateProgramCounter (
    size_t value )
```

Update the program counter.

Parameters

<i>value</i>	new value
--------------	-----------

9.43.3.28 updateRegValue()

```
void MainWindow::updateRegValue (
    int no,
    uint32_t value,
    const QBrush & brush = QBrush("red"),
    bool init = false )
```

Update the target register value.

Parameters

<i>no</i>	register number
<i>value</i>	new value
<i>brush</i>	the texture brush (to indicate there is a modification on this register)
<i>init</i>	whether this is a initialization (may lead to different behavior when updating the stack pointer)

9.43.3.29 updateStack()

```
void MainWindow::updateStack (
    uint32_t addr,
    size_t size )
```

Update the value in the target address in GUI.

Parameters

<i>addr</i>	address to update
<i>size</i>	number of bytes to update

9.43.4 Member Data Documentation**9.43.4.1 ACC**

```
union { ... } MainWindow::ACC
```

The accumulator for multiplication related operations

9.43.4.2 advanceCounter

```
bool MainWindow::advanceCounter = true
```

An indicator to check whether we should advance the counter or not. Used for jumping related instructions.

9.43.4.3 all

```
uint64_t MainWindow::all
```

stands for the whole accumulator

9.43.4.4 executor

```
Executor* MainWindow::executor = nullptr
```

The pointer to the executor.

9.43.4.5 frame

```
std::array<char, ( 1024 * KiB) > MainWindow::frame
```

9.43.4.6 heap

```
Heap MainWindow::heap
```

The heap operation simulator.

9.43.4.7 high

```
uint32_t MainWindow::high
```

stands for the high part of the accumulator

9.43.4.8 instructions

```
std::vector<Instruction> MainWindow::instructions {}
```

The vector for instruction storage.

9.43.4.9 low

```
uint32_t MainWindow::low
```

stands for the low part of the accumulator

9.43.4.10 part

```
struct { ... } ::LOW_HIGH MainWindow::part
```

9.43.4.11 PC

```
size_t MainWindow::PC = 0
```

The program counter.

9.43.4.12 REGS

```
uint32_t MainWindow::REGS[32] = {}
```

The array for all registers.

9.43.4.13 stack

```
Stack MainWindow::stack
```

The stack operation simulator.

9.43.4.14 timer

```
QTimer MainWindow::timer
```

A timer to ignite periodical signals on execution.

9.43.4.15 ui

```
Ui::MainWindow* MainWindow::ui
```

The pointer to the Qt UI components.

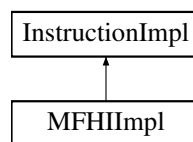
The documentation for this class was generated from the following files:

- [src/mainwindow.h](#)
- [src/mainwindow.cpp](#)
- [src/mainwindow.ipp](#)

9.44 MFHIImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for MFHIImpl:



Public Member Functions

- [MFHIImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.44.1 Constructor & Destructor Documentation

9.44.1.1 MFHIImpl()

```
MFHIImpl::MFHIImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.44.2 Member Function Documentation

9.44.2.1 exec()

```
void MFLOImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

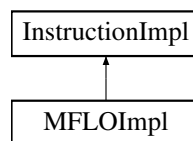
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.45 MFLOImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for MFLOImpl:



Public Member Functions

- [MFLOImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.45.1 Constructor & Destructor Documentation

9.45.1.1 MFLOImpl()

```
MFLOImpl::MFLOImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.45.2 Member Function Documentation

9.45.2.1 exec()

```
void MFLOImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

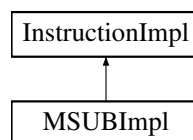
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.46 MSUBImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for MSUBImpl:



Public Member Functions

- [MSUBImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.46.1 Constructor & Destructor Documentation

9.46.1.1 MSUBImpl()

```
MSUBImpl::MSUBImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.46.2 Member Function Documentation

9.46.2.1 exec()

```
void MSUBUImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

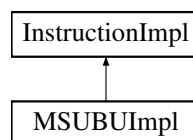
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.47 MSUBUImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for MSUBUImpl:



Public Member Functions

- [MSUBUImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.47.1 Constructor & Destructor Documentation

9.47.1.1 MSUBUImpl()

```
MSUBUImpl::MSUBUImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.47.2 Member Function Documentation

9.47.2.1 exec()

```
void MSUBUImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

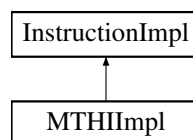
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.48 MTHImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for MTHImpl:



Public Member Functions

- [MTHImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.48.1 Constructor & Destructor Documentation

9.48.1.1 MTHImpl()

```
MTHImpl::MTHImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.48.2 Member Function Documentation

9.48.2.1 exec()

```
void MTHIImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

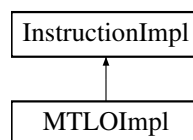
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.49 MTLOImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for MTLOImpl:



Public Member Functions

- [MTLOImpl](#) (Instruction *instr*)
- void [exec](#) () override

Additional Inherited Members

9.49.1 Constructor & Destructor Documentation

9.49.1.1 MTLOImpl()

```
MTLOImpl::MTLOImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.49.2 Member Function Documentation

9.49.2.1 exec()

```
void MTLOImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

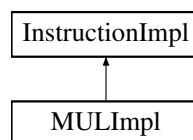
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.50 MULImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for MULImpl:



Public Member Functions

- [MULImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.50.1 Constructor & Destructor Documentation

9.50.1.1 MULImpl()

```
MULImpl::MULImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.50.2 Member Function Documentation

9.50.2.1 exec()

```
void MULTImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

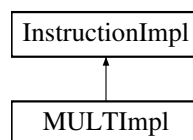
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.51 MULTImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for MULTImpl:



Public Member Functions

- [MULTImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.51.1 Constructor & Destructor Documentation

9.51.1.1 MULTImpl()

```
MULTImpl::MULTImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.51.2 Member Function Documentation

9.51.2.1 exec()

```
void MULTUImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

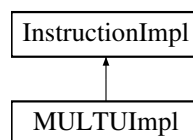
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.52 MULTUImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for MULTUImpl:



Public Member Functions

- [MULTUImpl](#) (Instruction *instr*)
- void [exec](#) () override

Additional Inherited Members

9.52.1 Constructor & Destructor Documentation

9.52.1.1 MULTUImpl()

```
MULTUImpl::MULTUImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.52.2 Member Function Documentation

9.52.2.1 exec()

```
void MULTUImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

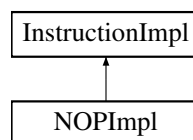
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.53 NOImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for NOImpl:



Public Member Functions

- [NOImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.53.1 Constructor & Destructor Documentation

9.53.1.1 NOImpl()

```
NOImpl::NOImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.53.2 Member Function Documentation

9.53.2.1 exec()

```
void NOPImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

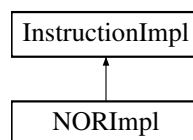
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.54 NORImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for NORImpl:



Public Member Functions

- [NORImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.54.1 Constructor & Destructor Documentation

9.54.1.1 NORImpl()

```
NORImpl::NORImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.54.2 Member Function Documentation

9.54.2.1 exec()

```
void ORIImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

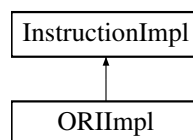
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.55 ORIImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for ORIImpl:



Public Member Functions

- [ORIImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.55.1 Constructor & Destructor Documentation

9.55.1.1 ORIImpl()

```
ORIImpl::ORIImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.55.2 Member Function Documentation

9.55.2.1 exec()

```
void ORIImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

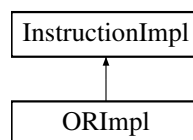
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.56 ORIImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for ORIImpl:



Public Member Functions

- [ORIImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.56.1 Constructor & Destructor Documentation

9.56.1.1 ORIImpl()

```
ORIImpl::ORIImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.56.2 Member Function Documentation

9.56.2.1 exec()

```
void ORImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

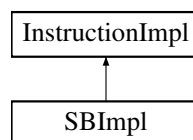
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.57 SBImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SBImpl:



Public Member Functions

- [SBImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.57.1 Constructor & Destructor Documentation

9.57.1.1 SBImpl()

```
SBImpl::SBImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.57.2 Member Function Documentation

9.57.2.1 exec()

```
void SBImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

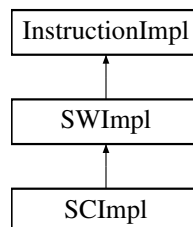
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.58 SCImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SCImpl:



Public Member Functions

- [SCImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.58.1 Constructor & Destructor Documentation

9.58.1.1 SCImpl()

```
SCImpl::SCImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.58.2 Member Function Documentation

9.58.2.1 exec()

```
void SCImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Attention

we have no difference between atomic ones and unatomic ones.

Reimplemented from [SWImpl](#).

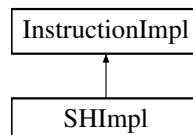
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.59 SHImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SHImpl:



Public Member Functions

- [SHImpl](#) (Instruction *instr*)
- void [exec](#) () override

Additional Inherited Members

9.59.1 Constructor & Destructor Documentation

9.59.1.1 SHImpl()

```
SHImpl::SHImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.59.2 Member Function Documentation

9.59.2.1 exec()

```
void SHImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

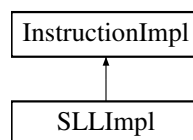
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.60 SLLImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SLLImpl:



Public Member Functions

- [SLLImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.60.1 Constructor & Destructor Documentation

9.60.1.1 SLLImpl()

```
SLLImpl::SLLImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.60.2 Member Function Documentation

9.60.2.1 exec()

```
void SLLVImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

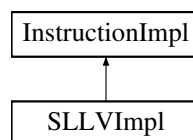
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.61 SLLVImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SLLVImpl:



Public Member Functions

- [SLLVImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.61.1 Constructor & Destructor Documentation

9.61.1.1 SLLVImpl()

```
SLLVImpl::SLLVImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.61.2 Member Function Documentation

9.61.2.1 exec()

```
void SLVImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

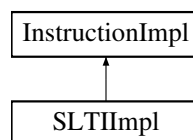
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.62 SLTImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SLTImpl:



Public Member Functions

- [SLTImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.62.1 Constructor & Destructor Documentation

9.62.1.1 SLTImpl()

```
SLTImpl::SLTImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.62.2 Member Function Documentation

9.62.2.1 exec()

```
void SLTImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

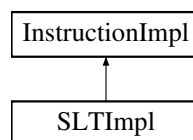
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.63 SLTImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SLTImpl:



Public Member Functions

- [SLTImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.63.1 Constructor & Destructor Documentation

9.63.1.1 SLTImpl()

```
SLTImpl::SLTImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.63.2 Member Function Documentation

9.63.2.1 exec()

```
void SLTIUImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

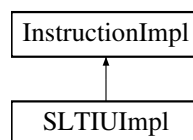
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.64 SLTIUImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SLTIUImpl:



Public Member Functions

- [SLTIUImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.64.1 Constructor & Destructor Documentation

9.64.1.1 SLTIUImpl()

```
SLTIUImpl::SLTIUImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.64.2 Member Function Documentation

9.64.2.1 exec()

```
void SLTUImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

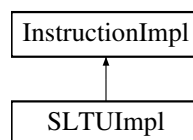
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.65 SLTUImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SLTUImpl:



Public Member Functions

- [SLTUImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.65.1 Constructor & Destructor Documentation

9.65.1.1 SLTUImpl()

```
SLTUImpl::SLTUImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.65.2 Member Function Documentation

9.65.2.1 exec()

```
void SLTUImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

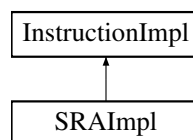
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.66 SRAImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SRAImpl:



Public Member Functions

- [SRAImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.66.1 Constructor & Destructor Documentation

9.66.1.1 SRAImpl()

```
SRAImpl::SRAImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.66.2 Member Function Documentation

9.66.2.1 exec()

```
void SRAVImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

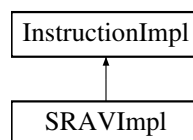
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.67 SRAVImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SRAVImpl:



Public Member Functions

- [SRAVImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.67.1 Constructor & Destructor Documentation

9.67.1.1 SRAVImpl()

```
SRAVImpl::SRAVImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.67.2 Member Function Documentation

9.67.2.1 exec()

```
void SRAVImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

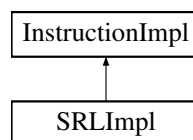
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.68 SRLImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SRLImpl:



Public Member Functions

- [SRLImpl](#) (Instruction *instr*)
- void [exec](#) () override

Additional Inherited Members

9.68.1 Constructor & Destructor Documentation

9.68.1.1 SRLImpl()

```
SRLImpl::SRLImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.68.2 Member Function Documentation

9.68.2.1 exec()

```
void SRLVImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

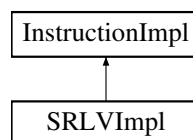
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.69 SRLVImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SRLVImpl:



Public Member Functions

- [SRLVImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.69.1 Constructor & Destructor Documentation

9.69.1.1 SRLVImpl()

```
SRLVImpl::SRLVImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.69.2 Member Function Documentation

9.69.2.1 exec()

```
void SRLVImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.70 Stack Struct Reference

```
#include <stack.h>
```

Public Member Functions

- [Stack](#) ()
- [~Stack](#) ()
- void [grow](#) (size_t scale)
- bool [isEnoughFor](#) (size_t target)
- size_t [size](#) () const
- template<class T >
T * [get](#) (uint32_t addr)
- void [enlarge](#) (size_t n)
- void [decrease](#) (size_t n)
- long [order](#) (uint32_t addr)
- void [clear](#) ()

Public Attributes

- size_t [capacity](#)
current capacity
- char * [highest](#)
memory area ending
- char * [current](#)
memory available address

9.70.1 Detailed Description

The stack simulator

9.70.2 Constructor & Destructor Documentation

9.70.2.1 Stack()

```
Stack::Stack ( )
```

Construct the stack. This will pre-allocate some memory and initialize the pointers

9.70.2.2 ~Stack()

```
Stack::~~Stack ( )
```

Destruct the stack. This will free the holding memory

9.70.3 Member Function Documentation

9.70.3.1 clear()

```
void Stack::clear ( )
```

Clear the stack. The behavior is quite similar to initialization

9.70.3.2 decrease()

```
void Stack::decrease (
    size_t n )
```

Decrease the stack size

Parameters

<i>n</i>	The size to decrease by
----------	-------------------------

9.70.3.3 enlarge()

```
void Stack::enlarge (
    size_t n )
```

Enlarge the stack size

Parameters

<i>n</i>	The size to enlarge by
----------	------------------------

9.70.3.4 get()

```
template<class T >
T * Stack::get (
    uint32_t addr )
```

Get the address at the simulated memory index

Template Parameters

<i>T</i>	data type
----------	-----------

Parameters

<i>addr</i>	target index
-------------	--------------

Returns

real address

9.70.3.5 grow()

```
void Stack::grow (
    size_t scale )
```

Enlarge the stack capacity

Parameters

<i>scale</i>	The capacity to enlarge to
--------------	----------------------------

9.70.3.6 isEnoughFor()

```
bool Stack::isEnoughFor (
    size_t target )
```

Check whether the current capacity is enough

Parameters

<i>target</i>	target size
---------------	-------------

Returns

the checking result

9.70.3.7 order()

```
long Stack::order (
    uint32_t addr )
```

Get the order of an address

Parameters

<i>addr</i>	target address
-------------	----------------

Returns

order

9.70.3.8 size()

```
size_t Stack::size ( ) const
```

The current size

Returns

The current size

9.70.4 Member Data Documentation

9.70.4.1 capacity

```
size_t Stack::capacity
```

current capacity

9.70.4.2 current

```
char* Stack::current
```

memory available address

9.70.4.3 highest

```
char* Stack::highest
```

memory area ending

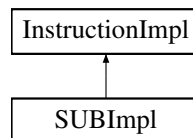
The documentation for this struct was generated from the following files:

- [src/stack.h](#)
- [src/stack.cpp](#)

9.71 SUBImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SUBImpl:



Public Member Functions

- [SUBImpl](#) (Instruction *instr*)
- void [exec](#) () override

Additional Inherited Members

9.71.1 Constructor & Destructor Documentation

9.71.1.1 SUBImpl()

```
SUBImpl::SUBImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.71.2 Member Function Documentation

9.71.2.1 exec()

```
void SUBUImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

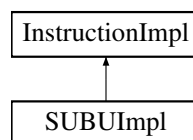
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.72 SUBUImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SUBUImpl:



Public Member Functions

- [SUBUImpl](#) (Instruction *instr*)
- void [exec](#) () override

Additional Inherited Members

9.72.1 Constructor & Destructor Documentation

9.72.1.1 SUBUImpl()

```
SUBUImpl::SUBUImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.72.2 Member Function Documentation

9.72.2.1 exec()

```
void SUBUImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

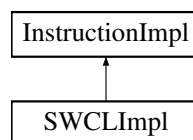
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.73 SWCLImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SWCLImpl:



Public Member Functions

- [SWCLImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.73.1 Constructor & Destructor Documentation

9.73.1.1 SWCLImpl()

```
SWCLImpl::SWCLImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.73.2 Member Function Documentation

9.73.2.1 exec()

```
void SWCLImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

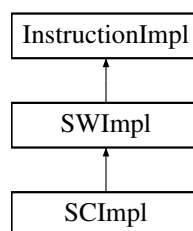
The documentation for this struct was generated from the following file:

- [src/instruction_impl.h](#)

9.74 SWImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SWImpl:



Public Member Functions

- [SWImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.74.1 Constructor & Destructor Documentation

9.74.1.1 SWImpl()

```
SWImpl::SWImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.74.2 Member Function Documentation

9.74.2.1 exec()

```
void SWImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

Reimplemented in [SCImpl](#).

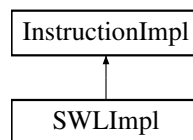
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.75 SWImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SWImpl:



Public Member Functions

- [SWImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.75.1 Constructor & Destructor Documentation

9.75.1.1 SWImpl()

```
SWImpl::SWImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.75.2 Member Function Documentation

9.75.2.1 exec()

```
void SWRImpI::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

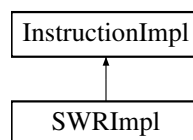
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.76 SWRImpI Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SWRImpI:



Public Member Functions

- [SWRImpI](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.76.1 Constructor & Destructor Documentation

9.76.1.1 SWRImpI()

```
SWRImpI::SWRImpI (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.76.2 Member Function Documentation

9.76.2.1 exec()

```
void SWRImp::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

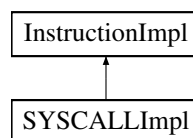
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.77 SYSCALLImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for SYSCALLImpl:



Public Member Functions

- [SYSCALLImpl](#) (Instruction *instr*)
- void [exec](#) () override

Additional Inherited Members

9.77.1 Constructor & Destructor Documentation

9.77.1.1 SYSCALLImpl()

```
SYSCALLImpl::SYSCALLImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.77.2 Member Function Documentation

9.77.2.1 exec()

```
void SYSCALLImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

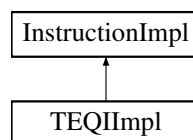
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.78 TEQIImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for TEQIImpl:



Public Member Functions

- [TEQIImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.78.1 Constructor & Destructor Documentation

9.78.1.1 TEQIImpl()

```
TEQIImpl::TEQIImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.78.2 Member Function Documentation

9.78.2.1 exec()

```
void TEQImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

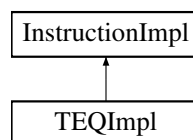
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.79 TEQImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for TEQImpl:



Public Member Functions

- [TEQImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.79.1 Constructor & Destructor Documentation

9.79.1.1 TEQImpl()

```
TEQImpl::TEQImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.79.2 Member Function Documentation

9.79.2.1 exec()

```
void TEQImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

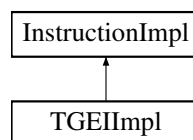
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.80 TGEImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for TGEImpl:



Public Member Functions

- [TGEImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.80.1 Constructor & Destructor Documentation

9.80.1.1 TGEImpl()

```
TGEImpl::TGEImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.80.2 Member Function Documentation

9.80.2.1 exec()

```
void TGEImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

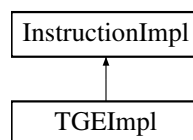
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.81 TGEImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for TGEImpl:



Public Member Functions

- [TGEImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.81.1 Constructor & Destructor Documentation

9.81.1.1 TGEImpl()

```
TGEImpl::TGEImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.81.2 Member Function Documentation

9.81.2.1 exec()

```
void TGEIUIImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

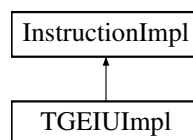
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.82 TGEIUIImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for TGEIUIImpl:



Public Member Functions

- [TGEIUIImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.82.1 Constructor & Destructor Documentation

9.82.1.1 TGEIUIImpl()

```
TGEIUIImpl::TGEIUIImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.82.2 Member Function Documentation

9.82.2.1 exec()

```
void TGEUImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

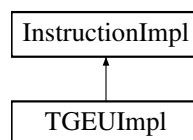
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.83 TGEUImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for TGEUImpl:



Public Member Functions

- [TGEUImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.83.1 Constructor & Destructor Documentation

9.83.1.1 TGEUImpl()

```
TGEUImpl::TGEUImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.83.2 Member Function Documentation

9.83.2.1 exec()

```
void TGEUImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

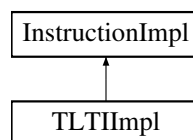
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.84 TLTIImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for TLTIImpl:



Public Member Functions

- [TLTIImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.84.1 Constructor & Destructor Documentation

9.84.1.1 TLTIImpl()

```
TLTIImpl::TLTIImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.84.2 Member Function Documentation

9.84.2.1 exec()

```
void TLTIImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

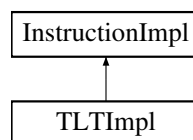
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.85 TLTIImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for TLTIImpl:



Public Member Functions

- [TLTIImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.85.1 Constructor & Destructor Documentation

9.85.1.1 TLTIImpl()

```
TLTIImpl::TLTIImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.85.2 Member Function Documentation

9.85.2.1 exec()

```
void TLTIUImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

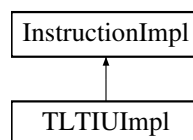
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.86 TLTIUImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for TLTIUImpl:



Public Member Functions

- [TLTIUImpl](#) (Instruction *instr*)
- void [exec](#) () override

Additional Inherited Members

9.86.1 Constructor & Destructor Documentation

9.86.1.1 TLTIUImpl()

```
TLTIUImpl::TLTIUImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.86.2 Member Function Documentation

9.86.2.1 exec()

```
void TLTUImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

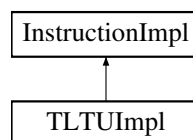
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.87 TLTUImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for TLTUImpl:



Public Member Functions

- [TLTUImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.87.1 Constructor & Destructor Documentation

9.87.1.1 TLTUImpl()

```
TLTUImpl::TLTUImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.87.2 Member Function Documentation

9.87.2.1 exec()

```
void TLTUImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

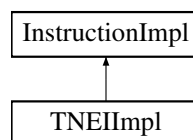
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.88 TNEImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for TNEImpl:



Public Member Functions

- [TNEImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.88.1 Constructor & Destructor Documentation

9.88.1.1 TNEImpl()

```
TNEImpl::TNEImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.88.2 Member Function Documentation

9.88.2.1 exec()

```
void TNEImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

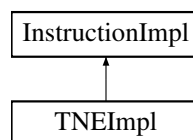
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.89 TNEImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for TNEImpl:



Public Member Functions

- [TNEImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.89.1 Constructor & Destructor Documentation

9.89.1.1 TNEImpl()

```
TNEImpl::TNEImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.89.2 Member Function Documentation

9.89.2.1 exec()

```
void TNEImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

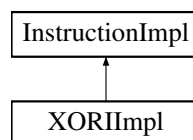
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.90 XORImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for XORImpl:



Public Member Functions

- [XORImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.90.1 Constructor & Destructor Documentation

9.90.1.1 XORImpl()

```
XORImpl::XORImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.90.2 Member Function Documentation

9.90.2.1 exec()

```
void XORImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

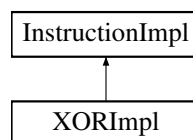
The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

9.91 XORImpl Struct Reference

```
#include <instruction_impl.h>
```

Inheritance diagram for XORImpl:



Public Member Functions

- [XORImpl](#) (Instruction [instr](#))
- void [exec](#) () override

Additional Inherited Members

9.91.1 Constructor & Destructor Documentation

9.91.1.1 XORImpl()

```
XORImpl::XORImpl (
    Instruction instr ) [explicit]
```

construct the target instruction implementation

Parameters

<i>instr</i>	the instruction value
--------------	-----------------------

9.91.2 Member Function Documentation

9.91.2.1 exec()

```
void XORImpl::exec ( ) [override], [virtual]
```

execute the instruction, depends on the real implementation in each struct

Implements [InstructionImpl](#).

The documentation for this struct was generated from the following files:

- [src/instruction_impl.h](#)
- [src/instruction_impl.cpp](#)

Chapter 10

File Documentation

10.1 README.md File Reference

10.2 REPORT.md File Reference

10.3 src/executor.cpp File Reference

```
#include "executor.h"  
#include <QDialog>  
#include <QVBoxLayout>  
#include <QLabel>  
#include <QPushButton>
```

10.4 src/executor.h File Reference

```
#include "instruction_impl.h"
```

Classes

- class [Executor](#)

10.5 src/fs.h File Reference

```
#include <sys/types.h>  
#include <sys/stat.h>  
#include <fcntl.h>  
#include <unistd.h>
```

10.6 src/global.cpp File Reference

```
#include "global.h"
```

Variables

- `size_t` [STATIC_HIGH](#) = 0
- `const char *` [REG_NAME](#) [32]
a mapping between register no to register name

10.6.1 Variable Documentation

10.6.1.1 REG_NAME

```
const char* REG_NAME[32]
```

Initial value:

```
= {
    "$zero", "$at", "$v0", "$v1", "$a0", "$a1", "$a2", "$a3", "$t0", "$t1", "$t2", "$t3", "$t4",
    "$t5", "$t6", "$t7", "$s0", "$s1", "$s2", "$s3", "$s4", "$s5", "$s6", "$s7", "$t8", "$t9",
    "$k0", "$k1", "$gp", "$sp", "$fp", "$ra"
}
```

a mapping between register no to register name

10.6.1.2 STATIC_HIGH

```
size_t STATIC_HIGH = 0
```

10.7 src/global.h File Reference

```
#include <cstddef>
#include <cstdint>
#include <memory>
#include <mimalloc.h>
```

Classes

- `struct` [_SIM::InstrDeleter](#)

Namespaces

- [_SIM](#)

Macros

- `#define` [BASE_ADDR](#) 0x000000u
start address of the MIPS program
- `#define` [STATIC_LOW](#) 0x500000u
- `#define` [LIKELY](#)(x) (x)
- `#define` [UNLIKELY](#)(x) (x)

Typedefs

- using [_SIM::InstrPtr](#) = std::unique_ptr< [InstructionImpl](#), InstrDeleter >

Functions

- template<typename T, typename ... Args>
InstrPtr [_SIM::make_unique](#) (Args &&...args)

Variables

- size_t [STATIC_HIGH](#)
- const char * [REG_NAME](#) [32]
a mapping between register no to register name

10.7.1 Macro Definition Documentation

10.7.1.1 BASE_ADDR

```
#define BASE_ADDR 0x000000u
```

start address of the MIPS program

10.7.1.2 LIKELY

```
#define LIKELY(  
    x ) (x)
```

10.7.1.3 STATIC_LOW

```
#define STATIC_LOW 0x500000u
```

10.7.1.4 UNLIKELY

```
#define UNLIKELY(  
    x ) (x)
```

10.7.2 Variable Documentation

10.7.2.1 REG_NAME

```
const char* REG_NAME[32]
```

a mapping between register no to register name

10.7.2.2 STATIC_HIGH

```
size_t STATIC_HIGH
```

10.8 src/heap.cpp File Reference

```
#include "heap.h"  
#include "global.h"  
#include <sys/mman.h>  
#include <sstream>
```

Functions

- void [segfault_sigaction](#) (int signal, siginfo_t *si, void *arg)
- void [bind_sigsegv](#) ()

10.8.1 Function Documentation

10.8.1.1 `bind_sigsegv()`

```
void bind_sigsegv ( )
```

Bind the signal SIGSEGV to our own handler.

10.8.1.2 `segfault_sigaction()`

```
void segfault_sigaction (
    int signal,
    siginfo_t * si,
    void * arg )
```

A segfault handler that is used to handle the invalid memory access

Parameters

<i>signal</i>	signal value
<i>si</i>	signal information
<i>arg</i>	arguments list

10.9 src/heap.h File Reference

```
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
#include <list>
#include <queue>
#include <mimalloc.h>
#include <csignal>
#include <cstring>
```

Classes

- class [Heap](#)

Typedefs

- template<class Key , class Value >
using [TreeSet](#) = tree< Key, Value, std::less< Key >, rb_tree_tag, tree_order_statistics_node_update >

Functions

- void [segfault_sigaction](#) (int signal, siginfo_t *si, void *arg)
- void [bind_sigsegv](#) ()

10.9.1 Typedef Documentation

10.9.1.1 TreeSet

```
template<class Key , class Value >
using TreeSet = tree<Key, Value, std::less<Key>, rb_tree_tag, tree_order_statistics_node_update>
```

A Policy Based Statistics Tree to help us keep the order of the allocated chunks

10.9.2 Function Documentation

10.9.2.1 bind_sigsegv()

```
void bind_sigsegv ( )
```

Bind the signal SIGSEGV to our own handler.

10.9.2.2 segfault_sigaction()

```
void segfault_sigaction (
    int signal,
    siginfo_t * si,
    void * arg )
```

A segfault handler that is used to handle the invalid memory access

Parameters

<i>signal</i>	signal value
<i>si</i>	signal information
<i>arg</i>	arguments list

10.10 src/instruction.cpp File Reference

```
#include "instruction.h"
#include <stdexcept>
```

Functions

- [TYPE resolv_type](#) (Instruction inst)

10.10.1 Function Documentation

10.10.1.1 resolv_type()

```
TYPE resolv_type (
    Instruction inst )
```

Detect the type of an raw instruction

Parameters

<i>inst</i>	the instruction to detect
-------------	---------------------------

Returns

the type of the instruction

10.11 src/instruction.h File Reference

```
#include "global.h"
#include <instructions_types.h>
```

Macros

- #define [OPC_J](#) 0b000010
- #define [OPC_JAL](#) 0b000011
- #define [OPC_ADDI](#) 0b001000
- #define [OPC_ADDIU](#) 0b001001
- #define [OPC_ANDI](#) 0b001100
- #define [OPC_BEQ](#) 0b000100
- #define [OPC_BGTZ](#) 0b000111
- #define [OPC_BLEZ](#) 0b000110
- #define [OPC_BNE](#) 0b000101
- #define [OPC_LB](#) 0b100000
- #define [OPC_LBU](#) 0b100100
- #define [OPC_LH](#) 0b100001
- #define [OPC_LHU](#) 0b100101
- #define [OPC_LUI](#) 0b001111
- #define [OPC_LW](#) 0b100011
- #define [OPC_ORI](#) 0b001101
- #define [OPC_SB](#) 0b101000
- #define [OPC_SLTI](#) 0b001010
- #define [OPC_SLTIU](#) 0b001011
- #define [OPC_SH](#) 0b101001
- #define [OPC_SW](#) 0b101011
- #define [OPC_SWCL](#) 0b111001

- #define `OPC_XORI` 0b001110
- #define `OPC_LWL` 0b100010
- #define `OPC_LWR` 0b100110
- #define `OPC_SWL` 0b101010
- #define `OPC_SWR` 0b101110
- #define `OPC_SC` 0b111000
- #define `OPC_LL` 0b110000
- #define `FCR_ADD` 0b100000
- #define `FCR_ADDU` 0b100001
- #define `FCR_AND` 0b100100
- #define `FCR_BREAK` 0b001101
- #define `FCR_DIV` 0b011010
- #define `FCR_DIVU` 0b011011
- #define `FCR_JALR` 0b001001
- #define `FCR_JR` 0b001000
- #define `FCR_MFHI` 0b010000
- #define `FCR_MFLO` 0b010010
- #define `FCR_MTHI` 0b010001
- #define `FCR_MTLO` 0b010011
- #define `FCR_MULT` 0b011000
- #define `FCR_MULTU` 0b011001
- #define `FCR_NOR` 0b100111
- #define `FCR_OR` 0b100101
- #define `FCR_SLL` 0b000000
- #define `FCR_SLLV` 0b000100
- #define `FCR_SLT` 0b101010
- #define `FCR_SLTU` 0b101011
- #define `FCR_SRA` 0b000011
- #define `FCR_SRAV` 0b000111
- #define `FCR_SRL` 0b000010
- #define `FCR_SRLV` 0b000110
- #define `FCR_SUB` 0b100010
- #define `FCR_SUBU` 0b100011
- #define `FCR_SYSCALL` 0b001100
- #define `FCR_XOR` 0b100110
- #define `FCR_TEQ` 0b110100
- #define `FCR_TNE` 0b110110
- #define `FCR_TGE` 0b110000
- #define `FCR_TGEU` 0b110001
- #define `FCR_TLT` 0b110010
- #define `FCR_TLTU` 0b110011
- #define `RI_BLTZ` 0b000000
- #define `RI_BGEZ` 0b000001
- #define `RI_TGEI` 0b010000
- #define `RI_TGEIU` 0b010001
- #define `RI_TLTI` 0b010100
- #define `RI_TLTIU` 0b010101
- #define `RI_TEQI` 0b011000
- #define `RI_TNEI` 0b011100
- #define `RI_BLTZAL` 0b100000
- #define `RI_BGEZAL` 0b100001
- #define `RLIKE_CLO` 0b100001
- #define `RLIKE_CLZ` 0b100000
- #define `RLIKE_MUL` 0b000010
- #define `RLIKE_MADD` 0b000000
- #define `RLIKE_MADDU` 0b000001
- #define `RLIKE_MSUB` 0b000100
- #define `RLIKE_MSUBU` 0b000101

Enumerations

- enum [TYPE](#) : uint8_t {
 [R](#), [I](#), [J](#), [RI](#),
 [RLIKE](#) }

Functions

- [TYPE](#) [resolv_type](#) (Instruction inst)

10.11.1 Macro Definition Documentation

10.11.1.1 FCR_ADD

```
#define FCR_ADD 0b100000
```

10.11.1.2 FCR_ADDU

```
#define FCR_ADDU 0b100001
```

10.11.1.3 FCR_AND

```
#define FCR_AND 0b100100
```

10.11.1.4 FCR_BREAK

```
#define FCR_BREAK 0b001101
```

10.11.1.5 FCR_DIV

```
#define FCR_DIV 0b011010
```

10.11.1.6 FCR_DIVU

```
#define FCR_DIVU 0b011011
```

10.11.1.7 FCR_JALR

```
#define FCR_JALR 0b001001
```

10.11.1.8 FCR_JR

```
#define FCR_JR 0b001000
```

10.11.1.9 FCR_MFHI

```
#define FCR_MFHI 0b010000
```

10.11.1.10 FCR_MFLO

```
#define FCR_MFLO 0b010010
```

10.11.1.11 FCR_MTHI

```
#define FCR_MTHI 0b010001
```

10.11.1.12 FCR_MTLO

```
#define FCR_MTLO 0b010011
```

10.11.1.13 FCR_MULT

```
#define FCR_MULT 0b011000
```

10.11.1.14 FCR_MULTU

```
#define FCR_MULTU 0b011001
```

10.11.1.15 FCR_NOR

```
#define FCR_NOR 0b100111
```

10.11.1.16 FCR_OR

```
#define FCR_OR 0b100101
```

10.11.1.17 FCR_SLL

```
#define FCR_SLL 0b000000
```

10.11.1.18 FCR_SLLV

```
#define FCR_SLLV 0b000100
```

10.11.1.19 FCR_SLT

```
#define FCR_SLT 0b101010
```

10.11.1.20 FCR_SLTU

```
#define FCR_SLTU 0b101011
```

10.11.1.21 FCR_SRA

```
#define FCR_SRA 0b000011
```

10.11.1.22 FCR_SRAV

```
#define FCR_SRAV 0b000111
```

10.11.1.23 FCR_SRL

```
#define FCR_SRL 0b000010
```

10.11.1.24 FCR_SRLV

```
#define FCR_SRLV 0b000110
```

10.11.1.25 FCR_SUB

```
#define FCR_SUB 0b100010
```

10.11.1.26 FCR_SUBU

```
#define FCR_SUBU 0b100011
```

10.11.1.27 FCR_SYSCALL

```
#define FCR_SYSCALL 0b001100
```

10.11.1.28 FCR_TEQ

```
#define FCR_TEQ 0b110100
```

10.11.1.29 FCR_TGE

```
#define FCR_TGE 0b110000
```

10.11.1.30 FCR_TGEU

```
#define FCR_TGEU 0b110001
```

10.11.1.31 FCR_TLT

```
#define FCR_TLT 0b110010
```

10.11.1.32 FCR_TLTU

```
#define FCR_TLTU 0b110011
```

10.11.1.33 FCR_TNE

```
#define FCR_TNE 0b110110
```

10.11.1.34 FCR_XOR

```
#define FCR_XOR 0b100110
```

10.11.1.35 OPC_ADDI

```
#define OPC_ADDI 0b001000
```

10.11.1.36 OPC_ADDIU

```
#define OPC_ADDIU 0b001001
```

10.11.1.37 OPC_ANDI

```
#define OPC_ANDI 0b001100
```

10.11.1.38 OPC_BEQ

```
#define OPC_BEQ 0b000100
```

10.11.1.39 OPC_BGTZ

```
#define OPC_BGTZ 0b000111
```

10.11.1.40 OPC_BLEZ

```
#define OPC_BLEZ 0b000110
```

10.11.1.41 OPC_BNE

```
#define OPC_BNE 0b000101
```

10.11.1.42 OPC_J

```
#define OPC_J 0b000010
```

10.11.1.43 OPC_JAL

```
#define OPC_JAL 0b000011
```

10.11.1.44 OPC_LB

```
#define OPC_LB 0b100000
```

10.11.1.45 OPC_LBU

```
#define OPC_LBU 0b100100
```


10.11.1.46 OPC_LH

```
#define OPC_LH 0b100001
```

10.11.1.47 OPC_LHU

```
#define OPC_LHU 0b100101
```

10.11.1.48 OPC_LL

```
#define OPC_LL 0b110000
```

10.11.1.49 OPC_LUI

```
#define OPC_LUI 0b001111
```

10.11.1.50 OPC_LW

```
#define OPC_LW 0b100011
```

10.11.1.51 OPC_LWL

```
#define OPC_LWL 0b100010
```

10.11.1.52 OPC_LWR

```
#define OPC_LWR 0b100110
```

10.11.1.53 OPC_ORI

```
#define OPC_ORI 0b001101
```

10.11.1.54 OPC_SB

```
#define OPC_SB 0b101000
```

10.11.1.55 OPC_SC

```
#define OPC_SC 0b111000
```

10.11.1.56 OPC_SH

```
#define OPC_SH 0b101001
```

10.11.1.57 OPC_SLTI

```
#define OPC_SLTI 0b001010
```

10.11.1.58 OPC_SLTIU

```
#define OPC_SLTIU 0b001011
```

10.11.1.59 OPC_SW

```
#define OPC_SW 0b101011
```

10.11.1.60 OPC_SWCL

```
#define OPC_SWCL 0b111001
```

10.11.1.61 OPC_SWL

```
#define OPC_SWL 0b101010
```

10.11.1.62 OPC_SWR

```
#define OPC_SWR 0b101110
```

10.11.1.63 OPC_XORI

```
#define OPC_XORI 0b001110
```

10.11.1.64 RI_BGEZ

```
#define RI_BGEZ 0b00001
```

10.11.1.65 RI_BGEZAL

```
#define RI_BGEZAL 0b10001
```

10.11.1.66 RI_BLTZ

```
#define RI_BLTZ 0b00000
```

10.11.1.67 RI_BLTZAL

```
#define RI_BLTZAL 0b10000
```

10.11.1.68 RI_TEQI

```
#define RI_TEQI 0b01100
```

10.11.1.69 RI_TGEI

```
#define RI_TGEI 0b01000
```

10.11.1.70 RI_TGEIU

```
#define RI_TGEIU 0b01001
```

10.11.1.71 RI_TLTI

```
#define RI_TLTI 0b01010
```

10.11.1.72 RI_TLTIU

```
#define RI_TLTIU 0b01011
```

10.11.1.73 RI_TNEI

```
#define RI_TNEI 0b01110
```

10.11.1.74 RLIKE_CLO

```
#define RLIKE_CLO 0b100001
```

10.11.1.75 RLIKE_CLZ

```
#define RLIKE_CLZ 0b100000
```

10.11.1.76 RLIKE_MADD

```
#define RLIKE_MADD 0b000000
```

10.11.1.77 RLIKE_MADDU

```
#define RLIKE_MADDU 0b000001
```

10.11.1.78 RLIKE_MSUB

```
#define RLIKE_MSUB 0b000100
```

10.11.1.79 RLIKE_MSUBU

```
#define RLIKE_MSUBU 0b000101
```

10.11.1.80 RLIKE_MUL

```
#define RLIKE_MUL 0b000010
```

10.11.2 Enumeration Type Documentation

10.11.2.1 TYPE

```
enum TYPE : uint8_t
```

An enumeration for different category of instructions

Enumerator

R	R-Type MIPS Instructions.
I	I-Type MIPS Instructions.
J	J-Type MIPS Instructions.
RI	RI-Type MIPS Instructions.
RLIKE	RLIKE-Type MIPS Instructions.

10.11.3 Function Documentation

10.11.3.1 resolv_type()

```
TYPE resolv_type (  
    Instruction inst )
```

Detect the type of an raw instruction

Parameters

<i>inst</i>	the instruction to detect
-------------	---------------------------

Returns

the type of the instruction

10.12 src/instruction_impl.cpp File Reference

```
#include "instruction_impl.h"
#include "mainwindow.ipp"
#include "syscall.h"
```

10.13 src/instruction_impl.h File Reference

```
#include "mainwindow.h"
#include "../ui_mainwindow.h"
```

Classes

- struct [InstructionImpl](#)
- struct [NOPImpl](#)
- struct [JImpl](#)
- struct [JALImpl](#)
- struct [ADDImpl](#)
- struct [ADDIUIImpl](#)
- struct [ANDImpl](#)
- struct [BEQImpl](#)
- struct [BGEZImpl](#)
- struct [BGTZImpl](#)
- struct [BLEZImpl](#)
- struct [BLTZImpl](#)
- struct [BNEImpl](#)
- struct [LBImpl](#)
- struct [LBUImpl](#)
- struct [LHImpl](#)
- struct [LHUImpl](#)
- struct [LUIImpl](#)
- struct [LWImpl](#)
- struct [ORIImpl](#)
- struct [SBIImpl](#)
- struct [SLTImpl](#)
- struct [SLTIUIImpl](#)
- struct [SHImpl](#)
- struct [SWImpl](#)
- struct [SWCImpl](#)

- struct [XORImpl](#)
- struct [ADDImpl](#)
- struct [ADDUImpl](#)
- struct [ANDImpl](#)
- struct [BREAKImpl](#)
- struct [DIVImpl](#)
- struct [DIVUImpl](#)
- struct [JRImpl](#)
- struct [JALRImpl](#)
- struct [MFHIImpl](#)
- struct [MFLOImpl](#)
- struct [MTHIImpl](#)
- struct [MTLOImpl](#)
- struct [MULTImpl](#)
- struct [MULTUImpl](#)
- struct [NORImpl](#)
- struct [ORImpl](#)
- struct [SLLImpl](#)
- struct [SLLVImpl](#)
- struct [SLTImpl](#)
- struct [SLTUImpl](#)
- struct [SRAImpl](#)
- struct [SRAVImpl](#)
- struct [SRLImpl](#)
- struct [SRLVImpl](#)
- struct [SUBImpl](#)
- struct [SUBUImpl](#)
- struct [SYSCALLImpl](#)
- struct [XORImpl](#)
- struct [BLTZLImpl](#)
- struct [BGEZLImpl](#)
- struct [TGEImpl](#)
- struct [TGEUImpl](#)
- struct [TLTIImpl](#)
- struct [TLTIUImpl](#)
- struct [TEQImpl](#)
- struct [TNEImpl](#)
- struct [BLTZALImpl](#)
- struct [BGEZALImpl](#)
- struct [BLTZALLImpl](#)
- struct [BGEZALLImpl](#)
- struct [CLOImpl](#)
- struct [CLZImpl](#)
- struct [MULImpl](#)
- struct [MADDImpl](#)
- struct [MADDUImpl](#)
- struct [MSUBImpl](#)
- struct [MSUBUImpl](#)
- struct [TEQImpl](#)
- struct [TNEImpl](#)
- struct [TGEImpl](#)
- struct [TGEUImpl](#)
- struct [TLTImpl](#)
- struct [TLTUImpl](#)
- struct [SWLImpl](#)

- struct [SWRImp](#)
- struct [LWLImp](#)
- struct [LWRImp](#)
- struct [SCLimp](#)
- struct [LLImp](#)

Macros

- `#define DEFAULT_INIT(NAME, FATHER) NAME##Impl::NAME##Impl(Instruction instr) : FATHER←
ER##Impl(instr) {}`
- `#define ComDef(CLASS, FATHER)`
- `#define SimDef(CLASS) ComDef(CLASS, Instruction)`
- `#define ComImplDef(CLASS, FATHER, BLOCK)`
- `#define SimImplDef(CLASS, BLOCK) ComImplDef(CLASS, Instruction, BLOCK)`
- `#define OP_AMONG_REGS(NAME, A, B, op, C)`
- `#define OP_AMONG_REGS_OVERFLOW(NAME, A, B, op, C)`
- `#define SHIFT_REAL(NAME, A, B, op, C, TYPE)`
- `#define SHIFT_IMM(NAME, A, B, op, C, TYPE)`
- `#define TRAP_R(NAME, op, TYPE)`
- `#define TRAP_RI(NAME, op, TYPE)`
- `#define BRANCH_IF(NAME, COND)`
- `#define BRANCH_IF_SAVE(NAME, COND)`

10.13.1 Macro Definition Documentation

10.13.1.1 BRANCH_IF

```
#define BRANCH_IF(  
    NAME,  
    COND )
```

Value:

```
SimImplDef(NAME, {\n  
    if (COND) {\n  
        mainW->updateProgramCounter(mainW->PC + (int16_t) instr.INST_I.C * int32_t(4));\n  
    }\n  
})
```

Generate the real definition of Branch instructions

10.13.1.2 BRANCH_IF_SAVE

```
#define BRANCH_IF_SAVE(  
    NAME,  
    COND )
```

Value:

```
SimImplDef(NAME, {\n  
    if (COND) {\n  
        mainW->updateRegValue(31, mainW->PC + 4);\n  
        mainW->updateProgramCounter(mainW->PC + (int16_t) instr.INST_I.C * int32_t(4));\n  
    }\n  
})
```

Generate the real definition of Branch instructions with save operation

10.13.1.3 ComDef

```
#define ComDef(
    CLASS,
    FATHER )
```

Value:

```
struct CLASS##Impl : public FATHER##Impl {\ \
    explicit CLASS##Impl( Instruction instr);\ \
    void exec() override;\
};
```

Complex declaration of an instruction implementation class.

Parameters

<i>NAME</i>	name of the instruction
<i>FATHER</i>	direct base of the instruction

10.13.1.4 ComImplDef

```
#define ComImplDef(
    CLASS,
    FATHER,
    BLOCK )
```

Value:

```
void CLASS##Impl::exec() BLOCK\
DEFAULT_INIT(CLASS, FATHER)
```

Complex real definition of instruction behaviors (custom direct base)

Attention

must be used together with declarations

10.13.1.5 DEFAULT_INIT

```
#define DEFAULT_INIT(
    NAME,
    FATHER ) NAME##Impl::NAME##Impl(Instruction instr) : FATHER##Impl(instr) {}
```

A handy macro to create a default constructor based on *NAME* and *FATHER*.

Parameters

<i>NAME</i>	name of the instruction
<i>FATHER</i>	direct base of the instruction

10.13.1.6 OP_AMONG_REGS

```
#define OP_AMONG_REGS(
    NAME,
    A,
    B,
    op,
    C )
```

Value:

```
SimImplDef(NAME, {\
    mainW->updateRegValue(instr.INST_R.A, mainW->REGS[instr.INST_R.B] op mainW->REGS[instr.INST_R.C]);\
})
```

Generate the real definition of the operations among registers

10.13.1.7 OP_AMONG_REGS_OVERFLOW

```
#define OP_AMONG_REGS_OVERFLOW(
    NAME,
    A,
    B,
    op,
    C )
```

Value:

```
SimImplDef(NAME, {\
    int32_t a = mainW->REGS[instr.INST_R.B];\
    int32_t b = mainW->REGS[instr.INST_R.C];\
    int32_t c = 0;\
    if(__builtin_#op##_overflow(a, b, &c)) throw std::runtime_error {"overflow"};\
    mainW->updateRegValue(instr.INST_R.A, c);\
})
```

Generate the real definition of the operations among registers with overflow checking

10.13.1.8 SHIFT_IMM

```
#define SHIFT_IMM(
    NAME,
    A,
    B,
    op,
    C,
    TYPE )
```

Value:

```
SimImplDef(NAME, {\
    mainW->updateRegValue(instr.INST_R.A,\
        static_cast<TYPE>(mainW->REGS[instr.INST_R.B]) op static_cast<TYPE>(instr.INST_R.C));\
})
```

Generate the real definition of shifting instructions whose shift among is stored in immediate value

10.13.1.9 SHIFT_REAL

```
#define SHIFT_REAL(
    NAME,
    A,
    B,
    op,
    C,
    TYPE )
```

Value:

```
SimImplDef(NAME, {\
    mainW->updateRegValue(instr.INST_R.A,\
        static_cast<TYPE>(mainW->REGS[instr.INST_R.B]) op static_cast<TYPE>(0b11111 &
        mainW->REGS[instr.INST_R.C]));\
})
```

Generate the real definition of shifting instructions whose shift among is stored in register

10.13.1.10 SimDef

```
#define SimDef(
    CLASS ) ComDef(CLASS, Instruction)
```

Simple declaration for those instruction implementations with a direct base of `InstructionImpl`

Parameters

<i>CLASS</i>	the name of the instruction
--------------	-----------------------------

10.13.1.11 SimImplDef

```
#define SimImplDef(
    CLASS,
    BLOCK ) ComImplDef(CLASS, Instruction, BLOCK)
```

Simple real definition of instruction behaviors (default direct base)

Attention

must be used together with declarations

10.13.1.12 TRAP_R

```
#define TRAP_R(
    NAME,
    OP,
    TYPE )
```

Value:

```
SimImplDef(NAME, {\
    TYPE a = mainW->REGS[instr.INST_R.s];\
    TYPE b = mainW->REGS[instr.INST_R.t];\
    if (a op b) throw std::runtime_error {"conditionally trapped!";\
})
```

Generate the real definition of R-Type Trap instructions

10.13.1.13 TRAP_RI

```
#define TRAP_RI(
    NAME,
    OP,
    TYPE )
```

Value:

```
SimImplDef(NAME, {\
    TYPE a = mainW->REGS[instr.INST_I.s];\
    TYPE b = instr.INST_I.C;\
    if (a op b) throw std::runtime_error {"conditionally trapped!";\
})
```

Generate the real definition of RI-Type Trap instructions

10.14 src/main.cpp File Reference

```
#include "mainwindow.h"
#include <QApplication>
```

Functions

- int [main](#) (int argc, char *argv[])

10.14.1 Function Documentation

10.14.1.1 main()

```
int main (
    int argc,
    char * argv[] )
```

10.15 src/mainwindow.cpp File Reference

```
#include "mainwindow.h"
#include "../ui_mainwindow.h"
#include "global.h"
#include <QMessageBox>
#include <QFileDialog>
#include <iostream>
#include <memory>
#include "instruction_impl.h"
#include "mainwindow.ipp"
#include "instruction.h"
#include <atomic>
#include "executor.h"
#include <cstring>
#include <QInputDialog>
#include "syscall.h"
#include "fs.h"
#include <QGraphicsView>
#include <QPainter>
```

10.16 src/mainwindow.h File Reference

```
#include <QMainWindow>
#include <QVector>
#include <QListWidget>
#include <memory>
#include <QThread>
#include <QTimer>
#include "instruction.h"
#include "heap.h"
#include "stack.h"
#include <assembler/include/api.h>
```

Classes

- class [MainWindow](#)

Namespaces

- [Ui](#)

Macros

- #define [KiB](#) 1024
- #define [MiB](#) ([KiB](#) * [KiB](#))
- #define [FRAME_SIZE](#) [MiB](#)
- #define [CASE](#)(NAME, [TYPE](#))
- #define [RCASE](#)(NAME) [CASE](#)(NAME, FCR)
- #define [IJCASE](#)(NAME) [CASE](#)(NAME, OPC)
- #define [RICASE](#)(NAME) [CASE](#)(NAME, RI)
- #define [RLCASE](#)(NAME) [CASE](#)(NAME, [RLIKE](#))
- #define [HANDLE](#)(NAME, BLOCK)

Enumerations

- enum `MemoryType` { `STATIC`, `HEAP`, `STACK` }

10.16.1 Macro Definition Documentation

10.16.1.1 CASE

```
#define CASE(  
    NAME,  
    TYPE )
```

Value:

```
case TYPE##_##NAME:\n    executor->impls[i] = _SIM::make_unique<NAME##Impl> (instr);\n    break;
```

Generate a branch case for instruction handling

10.16.1.2 FRAME_SIZE

```
#define FRAME_SIZE MiB
```

10.16.1.3 HANDLE

```
#define HANDLE(  
    NAME,  
    BLOCK )
```

Value:

```
case SYSCALL_##NAME:\n    BLOCK\n    break;
```

Generate a branch case for syscall handling

10.16.1.4 IJCASE

```
#define IJCASE(  
    NAME ) CASE(NAME, OPC)
```

Generate a branch case for I/J-Type instruction handling

10.16.1.5 KiB

```
#define KiB 1024
```

10.16.1.6 MiB

```
#define MiB (KiB * KiB)
```

10.16.1.7 RCASE

```
#define RCASE(  
    NAME ) CASE(NAME, FCR)
```

Generate a branch case for R-Type instruction handling

10.16.1.8 RICASE

```
#define RICASE(  
    NAME ) CASE(NAME, RI)
```

Generate a branch case for RI-Type instruction handling

10.16.1.9 RLCASE

```
#define RLCASE(  
    NAME ) CASE(NAME, RLIKE)
```

Generate a branch case for RLIKE-Type instruction handling

10.16.2 Enumeration Type Documentation

10.16.2.1 MemoryType

```
enum MemoryType
```

Different type of memories

Enumerator

STATIC	
HEAP	
STACK	

10.17 src/mainwindow.ipp File Reference

```
#include "../ui_mainwindow.h"  
#include "mainwindow.h"
```

Macros

- #define [MAINWINDOW_IPP](#)

10.17.1 Macro Definition Documentation

10.17.1.1 MAINWINDOW_IPP

```
#define MAINWINDOW_IPP
```

10.18 src/stack.cpp File Reference

```
#include "stack.h"
```

Functions

- uint32_t [nextPowerOfTwo](#) (uint32_t n)

10.18.1 Function Documentation

10.18.1.1 nextPowerOfTwo()

```
uint32_t nextPowerOfTwo (  
    uint32_t n )
```

10.19 src/stack.h File Reference

```
#include <cstddef>  
#include <mimalloc.h>  
#include <cstring>  
#include <stdexcept>
```


Classes

- struct [Stack](#)

Macros

- #define [DEFAULT_SIZE](#) 1u
- #define [STACK_HIGH](#) 0x7ffffffu

Functions

- uint32_t [nextPowerOfTwo](#) (uint32_t n)

10.19.1 Macro Definition Documentation

10.19.1.1 DEFAULT_SIZE

```
#define DEFAULT_SIZE 1u
```

10.19.1.2 STACK_HIGH

```
#define STACK_HIGH 0x7ffffffu
```

10.19.2 Function Documentation

10.19.2.1 nextPowerOfTwo()

```
uint32_t nextPowerOfTwo (  
    uint32_t n )
```

10.20 src/syscall.h File Reference

Macros

- #define SYSCALL_PRINT_CHAR 11
- #define SYSCALL_PRINT_INT 1
- #define SYSCALL_PRINT_STRING 4
- #define SYSCALL_READ_CHAR 12
- #define SYSCALL_READ_INT 5
- #define SYSCALL_READ_STRING 8
- #define SYSCALL_MMAP 9
- #define SYSCALL_EXIT 10
- #define SYSCALL_OPEN 13
- #define SYSCALL_READ 14
- #define SYSCALL_WRITE 15
- #define SYSCALL_CLOSE 16
- #define SYSCALL_EXIT2 17
- #define SYSCALL_FAST_COPY 10000
- #define SYSCALL_UI_OPEN_FILE 10001
- #define SYSCALL_MUNMAP 10002

10.20.1 Macro Definition Documentation

10.20.1.1 SYSCALL_CLOSE

```
#define SYSCALL_CLOSE 16
```

10.20.1.2 SYSCALL_EXIT

```
#define SYSCALL_EXIT 10
```

10.20.1.3 SYSCALL_EXIT2

```
#define SYSCALL_EXIT2 17
```

10.20.1.4 SYSCALL_FAST_COPY

```
#define SYSCALL_FAST_COPY 10000
```

10.20.1.5 SYSCALL_MMAP

```
#define SYSCALL_MMAP 9
```

10.20.1.6 SYSCALL_MUNMAP

```
#define SYSCALL_MUNMAP 10002
```

10.20.1.7 SYSCALL_OPEN

```
#define SYSCALL_OPEN 13
```

10.20.1.8 SYSCALL_PRINT_CHAR

```
#define SYSCALL_PRINT_CHAR 11
```

10.20.1.9 SYSCALL_PRINT_INT

```
#define SYSCALL_PRINT_INT 1
```

10.20.1.10 SYSCALL_PRINT_STRING

```
#define SYSCALL_PRINT_STRING 4
```

10.20.1.11 SYSCALL_READ

```
#define SYSCALL_READ 14
```

10.20.1.12 SYSCALL_READ_CHAR

```
#define SYSCALL_READ_CHAR 12
```

10.20.1.13 SYSCALL_READ_INT

```
#define SYSCALL_READ_INT 5
```

10.20.1.14 SYSCALL_READ_STRING

```
#define SYSCALL_READ_STRING 8
```

10.20.1.15 SYSCALL_UI_OPEN_FILE

```
#define SYSCALL_UI_OPEN_FILE 10001
```

10.20.1.16 SYSCALL_WRITE

```
#define SYSCALL_WRITE 15
```

Index

- [_SIM, 17](#)
 - [InstrPtr, 17](#)
 - [make_unique, 18](#)
 - [_SIM::InstrDeleter, 48](#)
 - [operator\(\), 49](#)
 - [~Heap](#)
 - [Heap, 47](#)
 - [~MainWindow](#)
 - [MainWindow, 70](#)
 - [~Stack](#)
 - [Stack, 109](#)
- [ACC](#)
 - [MainWindow, 79](#)
- [ADDImpl, 21](#)
 - [ADDImpl, 21](#)
 - [exec, 22](#)
- [ADDImpl, 22](#)
 - [ADDImpl, 22](#)
 - [exec, 23](#)
- [ADDIImpl, 23](#)
 - [ADDIImpl, 23](#)
 - [exec, 24](#)
- [ADDUImpl, 24](#)
 - [ADDUImpl, 24](#)
 - [exec, 25](#)
- [advanceCounter](#)
 - [MainWindow, 79](#)
- [all](#)
 - [MainWindow, 79](#)
- [alloc](#)
 - [Heap, 47](#)
- [allocHeap](#)
 - [MainWindow, 71](#)
- [ANDImpl, 25](#)
 - [ANDImpl, 25](#)
 - [exec, 26](#)
- [ANDImpl, 26](#)
 - [ANDImpl, 26](#)
 - [exec, 27](#)
- [BASE_ADDR](#)
 - [global.h, 137](#)
- [BEQImpl, 27](#)
 - [BEQImpl, 27](#)
 - [exec, 28](#)
- [BGEZALImpl, 28](#)
 - [BGEZALImpl, 28](#)
 - [exec, 29](#)
- [BGEZALLImpl, 29](#)
 - [BGEZALLImpl, 29](#)
 - [exec, 30](#)
- [BGEZImpl, 30](#)
 - [BGEZImpl, 30](#)
 - [exec, 31](#)
- [BGEZLImpl, 31](#)
 - [BGEZLImpl, 31](#)
 - [exec, 32](#)
- [BGTZImpl, 32](#)
 - [BGTZImpl, 32](#)
 - [exec, 33](#)
- [bind_sigsegv](#)
 - [heap.cpp, 138](#)
 - [heap.h, 140](#)
- [BLEZImpl, 33](#)
 - [BLEZImpl, 33](#)
 - [exec, 34](#)
- [BLTZALImpl, 34](#)
 - [BLTZALImpl, 34](#)
 - [exec, 35](#)
- [BLTZALLImpl, 35](#)
 - [BLTZALLImpl, 35](#)
 - [exec, 36](#)
- [BLTZImpl, 36](#)
 - [BLTZImpl, 36](#)
 - [exec, 37](#)
- [BLTZLImpl, 37](#)
 - [BLTZLImpl, 37](#)
 - [exec, 38](#)
- [BNEImpl, 38](#)
 - [BNEImpl, 38](#)
 - [exec, 39](#)
- [BRANCH_IF](#)
 - [instruction_impl.h, 156](#)
- [BRANCH_IF_SAVE](#)
 - [instruction_impl.h, 156](#)
- [BREAKImpl, 39](#)
 - [BREAKImpl, 39](#)
 - [exec, 40](#)
- [capacity](#)
 - [Stack, 111](#)
- [CASE](#)
 - [mainwindow.h, 162](#)
- [clear](#)
 - [Heap, 47](#)
 - [Stack, 109](#)
- [CLOImpl, 40](#)
 - [CLOImpl, 40](#)
 - [exec, 41](#)

- CLZImpl, 41
 - CLZImpl, 41
 - exec, 42
- ComDef
 - instruction_impl.h, 156
- ComImplDef
 - instruction_impl.h, 157
- current
 - Stack, 112
- dealloc
 - Heap, 47
- deallocHeap
 - MainWindow, 71
- decrease
 - Stack, 109
- decreaseStack
 - MainWindow, 72
- DEFAULT_INIT
 - instruction_impl.h, 157
- DEFAULT_SIZE
 - stack.h, 165
- DIVImpl, 42
 - DIVImpl, 42
 - exec, 43
- DIVUImpl, 43
 - DIVUImpl, 43
 - exec, 44
- edit
 - MainWindow, 72
- editHeap
 - MainWindow, 72
- editStack
 - MainWindow, 73
- enlarge
 - Stack, 109
- exec
 - ADDImpl, 22
 - ADDImpl, 23
 - ADDIImpl, 24
 - ADDUImpl, 25
 - ANDImpl, 26
 - ANDImpl, 27
 - BEQImpl, 28
 - BGEZALImpl, 29
 - BGEZALLImpl, 30
 - BGEZImpl, 31
 - BGEZLImpl, 32
 - BGTZImpl, 33
 - BLEZImpl, 34
 - BLTZALImpl, 35
 - BLTZALLImpl, 36
 - BLTZImpl, 37
 - BLTZLImpl, 38
 - BNEImpl, 39
 - BREAKImpl, 40
 - CLOImpl, 41
 - CLZImpl, 42
 - DIVImpl, 43
 - DIVUImpl, 44
 - InstructionImpl, 50
 - JALImpl, 52
 - JALRImpl, 54
 - JImpl, 55
 - JRImpl, 57
 - LBImpl, 58
 - LBUImpl, 59
 - LHImpl, 60
 - LHUImpl, 61
 - LLImpl, 62
 - LUIImpl, 63
 - LWImpl, 64
 - LWLImpl, 65
 - LWRImpl, 66
 - MADDImpl, 67
 - MADDUImpl, 68
 - MFHIImpl, 83
 - MFLOImpl, 84
 - MSUBImpl, 85
 - MSUBUImpl, 86
 - MTHIImpl, 87
 - MTLOImpl, 88
 - MULImpl, 89
 - MULTImpl, 90
 - MULTUImpl, 91
 - NOPIImpl, 92
 - NORIImpl, 93
 - ORIImpl, 94
 - ORIImpl, 95
 - SBIImpl, 96
 - SCImpl, 97
 - SHImpl, 98
 - SLLImpl, 99
 - SLLVImpl, 100
 - SLTIImpl, 101
 - SLTIImpl, 102
 - SLTIUImpl, 103
 - SLTUImpl, 104
 - SRAImpl, 105
 - SRAVImpl, 106
 - SRLImpl, 107
 - SRLVImpl, 108
 - SUBImpl, 113
 - SUBUImpl, 114
 - SWCLImpl, 115
 - SWImpl, 116
 - SWLImpl, 117
 - SWRImpl, 118
 - SYSCALLImpl, 119
 - TEQIImpl, 120
 - TEQImpl, 121
 - TGEIImpl, 122
 - TGEImpl, 123
 - TGEIImpl, 124
 - TGEUImpl, 125
 - TLTIImpl, 126

- TLTImpl, 127
- TLTIUImpl, 128
- TLTUImpl, 129
- TNEImpl, 130
- TNEImpl, 131
- XORIImpl, 132
- XORIImpl, 133
- Executor, 44
 - exit, 45
 - finished, 45
 - impls, 45
 - mainW, 46
 - next, 45
- executor
 - MainWindow, 80
- exit
 - Executor, 45
- FCR_ADD
 - instruction.h, 143
- FCR_ADDU
 - instruction.h, 143
- FCR_AND
 - instruction.h, 143
- FCR_BREAK
 - instruction.h, 143
- FCR_DIV
 - instruction.h, 143
- FCR_DIVU
 - instruction.h, 143
- FCR_JALR
 - instruction.h, 144
- FCR_JR
 - instruction.h, 144
- FCR_MFHI
 - instruction.h, 144
- FCR_MFLO
 - instruction.h, 144
- FCR_MTHI
 - instruction.h, 144
- FCR_MTLO
 - instruction.h, 144
- FCR_MULT
 - instruction.h, 144
- FCR_MULTU
 - instruction.h, 144
- FCR_NOR
 - instruction.h, 145
- FCR_OR
 - instruction.h, 145
- FCR_SLL
 - instruction.h, 145
- FCR_SLLV
 - instruction.h, 145
- FCR_SLT
 - instruction.h, 145
- FCR_SLTU
 - instruction.h, 145
- FCR_SRA
 - instruction.h, 145
- FCR_SRAV
 - instruction.h, 145
- FCR_SRL
 - instruction.h, 146
- FCR_SRLV
 - instruction.h, 146
- FCR_SUB
 - instruction.h, 146
- FCR_SUBU
 - instruction.h, 146
- FCR_SYSCALL
 - instruction.h, 146
- FCR_TEQ
 - instruction.h, 146
- FCR_TGE
 - instruction.h, 146
- FCR_TGEU
 - instruction.h, 146
- FCR_TLT
 - instruction.h, 147
- FCR_TLTU
 - instruction.h, 147
- FCR_TNE
 - instruction.h, 147
- FCR_XOR
 - instruction.h, 147
- fetchHeap
 - MainWindow, 73
- fetchStack
 - MainWindow, 73
- finished
 - Executor, 45
- frame
 - MainWindow, 80
- FRAME_SIZE
 - mainwindow.h, 162
- get
 - Stack, 110
- getRealAddr
 - MainWindow, 74
- global.cpp
 - REG_NAME, 136
 - STATIC_HIGH, 136
- global.h
 - BASE_ADDR, 137
 - LIKELY, 137
 - REG_NAME, 138
 - STATIC_HIGH, 138
 - STATIC_LOW, 137
 - UNLIKELY, 138
- grow
 - Stack, 110
- HANDLE
 - mainwindow.h, 162
- handleSyscall
 - MainWindow, 74

- HEAP
 - mainwindow.h, 163
- Heap, 46
 - ~Heap, 47
 - alloc, 47
 - clear, 47
 - dealloc, 47
 - mapping, 48
 - order, 48
 - size, 48
- heap
 - MainWindow, 80
- heap.cpp
 - bind_sigsegv, 138
 - segfault_sigaction, 139
- heap.h
 - bind_sigsegv, 140
 - segfault_sigaction, 140
 - TreeSet, 140
- high
 - MainWindow, 80
- highest
 - Stack, 112
- I
 - instruction.h, 153
- IJCASE
 - mainwindow.h, 162
- impls
 - Executor, 45
- increaseStack
 - MainWindow, 74
- instr
 - InstructionImpl, 51
- InstrPtr
 - _SIM, 17
- instruction.cpp
 - resolv_type, 141
- instruction.h
 - FCR_ADD, 143
 - FCR_ADDU, 143
 - FCR_AND, 143
 - FCR_BREAK, 143
 - FCR_DIV, 143
 - FCR_DIVU, 143
 - FCR_JALR, 144
 - FCR_JR, 144
 - FCR_MFHI, 144
 - FCR_MFLO, 144
 - FCR_MTHI, 144
 - FCR_MTLO, 144
 - FCR_MULT, 144
 - FCR_MULTU, 144
 - FCR_NOR, 145
 - FCR_OR, 145
 - FCR_SLL, 145
 - FCR_SLLV, 145
 - FCR_SLT, 145
 - FCR_SLTU, 145
 - FCR_SRA, 145
 - FCR_SRAV, 145
 - FCR_SRL, 146
 - FCR_SRLV, 146
 - FCR_SUB, 146
 - FCR_SUBU, 146
 - FCR_SYSCALL, 146
 - FCR_TEQ, 146
 - FCR_TGE, 146
 - FCR_TGEU, 146
 - FCR_TLT, 147
 - FCR_TLTU, 147
 - FCR_TNE, 147
 - FCR_XOR, 147
 - I, 153
 - J, 153
 - OPC_ADDI, 147
 - OPC_ADDIU, 147
 - OPC_ANDI, 147
 - OPC_BEQ, 147
 - OPC_BGTZ, 148
 - OPC_BLEZ, 148
 - OPC_BNE, 148
 - OPC_J, 148
 - OPC_JAL, 148
 - OPC_LB, 148
 - OPC_LBU, 148
 - OPC_LH, 148
 - OPC_LHU, 149
 - OPC_LL, 149
 - OPC_LUI, 149
 - OPC_LW, 149
 - OPC_LWL, 149
 - OPC_LWR, 149
 - OPC_ORI, 149
 - OPC_SB, 149
 - OPC_SC, 150
 - OPC_SH, 150
 - OPC_SLTI, 150
 - OPC_SLTIU, 150
 - OPC_SW, 150
 - OPC_SWCL, 150
 - OPC_SWL, 150
 - OPC_SWR, 150
 - OPC_XORI, 151
 - R, 153
 - resolv_type, 153
 - RI, 153
 - RI_BGEZ, 151
 - RI_BGEZAL, 151
 - RI_BLTZ, 151
 - RI_BLTZAL, 151
 - RI_TEQI, 151
 - RI_TGEI, 151
 - RI_TGEIU, 151
 - RI_TLTI, 152
 - RI_TLTIU, 152
 - RI_TNEI, 152

- RLIKE, 153
- RLIKE_CLO, 152
- RLIKE_CLZ, 152
- RLIKE_MADD, 152
- RLIKE_MADDU, 152
- RLIKE_MSUB, 152
- RLIKE_MSUBU, 153
- RLIKE_MUL, 153
- TYPE, 153
- instruction_impl.h
 - BRANCH_IF, 156
 - BRANCH_IF_SAVE, 156
 - ComDef, 156
 - ComImplDef, 157
 - DEFAULT_INIT, 157
 - OP_AMONG_REGS, 158
 - OP_AMONG_REGS_OVERFLOW, 158
 - SHIFT_IMM, 158
 - SHIFT_REAL, 158
 - SimDef, 159
 - SimImplDef, 159
 - TRAP_R, 159
 - TRAP_RI, 160
- InstructionImpl, 49
 - exec, 50
 - instr, 51
 - InstructionImpl, 50
 - mainW, 51
- instructions
 - MainWindow, 80
- isEnoughFor
 - Stack, 110
- J
 - instruction.h, 153
- JALImpl, 52
 - exec, 52
- JALImpl, 52
- JALRImpl, 53
 - exec, 54
- JALRImpl, 53
- JImpl, 54
 - exec, 55
- JImpl, 54
- JRImpl, 55
 - exec, 57
- JRImpl, 56
- KiB
 - mainwindow.h, 162
- LBImp, 57
 - exec, 58
- LBImp, 57
- LBUImpl, 58
 - exec, 59
- LBUImpl, 58
- LHImpl, 59
 - exec, 60
- LHImpl, 59
- LHUImp, 60
 - exec, 61
- LHUImp, 60
- LIKELY
 - global.h, 137
- LLImpl, 61
 - exec, 62
- LLImpl, 61
- low
 - MainWindow, 80
- LUIImpl, 62
 - exec, 63
- LUIImpl, 62
- LWImpl, 63
 - exec, 64
- LWImpl, 63
- LWLImpl, 64
 - exec, 65
- LWLImpl, 64
- LWRImpl, 65
 - exec, 66
- LWRImpl, 65
- MADDImpl, 66
 - exec, 67
- MADDImpl, 66
- MADDUImpl, 67
 - exec, 68
- MADDUImpl, 67
- main
 - main.cpp, 160
- main.cpp
 - main, 160
- mainW
 - Executor, 46
 - InstructionImpl, 51
- MainWindow, 68
 - ~MainWindow, 70
 - ACC, 79
 - advanceCounter, 79
 - all, 79
 - allocHeap, 71
 - deallocHeap, 71
 - decreaseStack, 72
 - edit, 72
 - editHeap, 72
 - editStack, 73
 - executor, 80
 - fetchHeap, 73
 - fetchStack, 73
 - frame, 80
 - getRealAddr, 74
 - handleSyscall, 74
 - heap, 80
 - high, 80
 - increaseStack, 74
 - instructions, 80
 - low, 80

- MainWindow, 70
- memoryType, 75
- on_aboutButton_clicked, 75
- on_executeButton_clicked, 75
- on_openButton_clicked, 75
- on_pushButton_clicked, 76
- on_resetButton_clicked, 76
- on_stepButton_clicked, 76
- on_stopButton_clicked, 76
- on_translateButton_clicked, 76
- part, 81
- PC, 81
- REGS, 81
- resetAll, 77
- showWarning, 77
- stack, 81
- timer, 81
- translateAll, 77
- ui, 81
- updateAcc, 77
- updateHigh, 78
- updateLow, 78
- updateProgramCounter, 78
- updateRegValue, 78
- updateStack, 79
- mainwindow.h
 - CASE, 162
 - FRAME_SIZE, 162
 - HANDLE, 162
 - HEAP, 163
 - IJCASE, 162
 - KiB, 162
 - MemoryType, 163
 - MiB, 162
 - RCASE, 163
 - RICASE, 163
 - RLCASE, 163
 - STACK, 163
 - STATIC, 163
- mainwindow.ipp
 - MAINWINDOW_IPP, 164
- MAINWINDOW_IPP
 - mainwindow.ipp, 164
- make_unique
 - _SIM, 18
- mapping
 - Heap, 48
- MemoryType
 - mainwindow.h, 163
- memoryType
 - MainWindow, 75
- MFHImpl, 82
 - exec, 83
 - MFHImpl, 82
- MFLOImpl, 83
 - exec, 84
 - MFLOImpl, 83
- MiB
 - mainwindow.h, 162
- MSUBImpl, 84
 - exec, 85
 - MSUBImpl, 84
- MSUBUImpl, 85
 - exec, 86
 - MSUBUImpl, 85
- MTHImpl, 86
 - exec, 87
 - MTHImpl, 86
- MTLOImpl, 87
 - exec, 88
 - MTLOImpl, 87
- MULImpl, 88
 - exec, 89
 - MULImpl, 88
- MULTImpl, 89
 - exec, 90
 - MULTImpl, 89
- MULTUImpl, 90
 - exec, 91
 - MULTUImpl, 90
- next
 - Executor, 45
- nextPowerOfTwo
 - stack.cpp, 164
 - stack.h, 165
- NOPIImpl, 91
 - exec, 92
 - NOPIImpl, 91
- NORIImpl, 92
 - exec, 93
 - NORIImpl, 92
- on_aboutButton_clicked
 - MainWindow, 75
- on_executeButton_clicked
 - MainWindow, 75
- on_openButton_clicked
 - MainWindow, 75
- on_pushButton_clicked
 - MainWindow, 76
- on_resetButton_clicked
 - MainWindow, 76
- on_stepButton_clicked
 - MainWindow, 76
- on_stopButton_clicked
 - MainWindow, 76
- on_translateButton_clicked
 - MainWindow, 76
- OP_AMONG_REGS
 - instruction_impl.h, 158
- OP_AMONG_REGS_OVERFLOW
 - instruction_impl.h, 158
- OPC_ADDI
 - instruction.h, 147
- OPC_ADDIU
 - instruction.h, 147

- OPC_ANDI
 - instruction.h, [147](#)
- OPC_BEQ
 - instruction.h, [147](#)
- OPC_BGTZ
 - instruction.h, [148](#)
- OPC_BLEZ
 - instruction.h, [148](#)
- OPC_BNE
 - instruction.h, [148](#)
- OPC_J
 - instruction.h, [148](#)
- OPC_JAL
 - instruction.h, [148](#)
- OPC_LB
 - instruction.h, [148](#)
- OPC_LBU
 - instruction.h, [148](#)
- OPC_LH
 - instruction.h, [148](#)
- OPC_LHU
 - instruction.h, [149](#)
- OPC_LL
 - instruction.h, [149](#)
- OPC_LUI
 - instruction.h, [149](#)
- OPC_LW
 - instruction.h, [149](#)
- OPC_LWL
 - instruction.h, [149](#)
- OPC_LWR
 - instruction.h, [149](#)
- OPC_ORI
 - instruction.h, [149](#)
- OPC_SB
 - instruction.h, [149](#)
- OPC_SC
 - instruction.h, [150](#)
- OPC_SH
 - instruction.h, [150](#)
- OPC_SLTI
 - instruction.h, [150](#)
- OPC_SLTIU
 - instruction.h, [150](#)
- OPC_SW
 - instruction.h, [150](#)
- OPC_SWCL
 - instruction.h, [150](#)
- OPC_SWL
 - instruction.h, [150](#)
- OPC_SWR
 - instruction.h, [150](#)
- OPC_XORI
 - instruction.h, [151](#)
- operator()
 - _SIM::InstrDeleter, [49](#)
- order
 - Heap, [48](#)
 - Stack, [111](#)
- ORImpl, [93](#)
 - exec, [94](#)
 - ORImpl, [93](#)
- ORImpl, [94](#)
 - exec, [95](#)
 - ORImpl, [94](#)
- part
 - MainWindow, [81](#)
- PC
 - MainWindow, [81](#)
- R
 - instruction.h, [153](#)
- RCASE
 - mainwindow.h, [163](#)
- README.md, [135](#)
- REG_NAME
 - global.cpp, [136](#)
 - global.h, [138](#)
- REGS
 - MainWindow, [81](#)
- REPORT.md, [135](#)
- resetAll
 - MainWindow, [77](#)
- resolv_type
 - instruction.cpp, [141](#)
 - instruction.h, [153](#)
- RI
 - instruction.h, [153](#)
- RI_BGEZ
 - instruction.h, [151](#)
- RI_BGEZAL
 - instruction.h, [151](#)
- RI_BLTZ
 - instruction.h, [151](#)
- RI_BLTZAL
 - instruction.h, [151](#)
- RI_TEQI
 - instruction.h, [151](#)
- RI_TGEI
 - instruction.h, [151](#)
- RI_TGEIU
 - instruction.h, [151](#)
- RI_TLTI
 - instruction.h, [152](#)
- RI_TLTIU
 - instruction.h, [152](#)
- RI_TNEI
 - instruction.h, [152](#)
- RICASE
 - mainwindow.h, [163](#)
- RLCASE
 - mainwindow.h, [163](#)
- RLIKE
 - instruction.h, [153](#)
- RLIKE_CLO
 - instruction.h, [152](#)

- RLIKE_CLZ
 - instruction.h, 152
- RLIKE_MADD
 - instruction.h, 152
- RLIKE_MADDU
 - instruction.h, 152
- RLIKE_MSUB
 - instruction.h, 152
- RLIKE_MSUBU
 - instruction.h, 153
- RLIKE_MUL
 - instruction.h, 153
- SBImp, 95
 - exec, 96
 - SBImp, 95
- SCImp, 96
 - exec, 97
 - SCImp, 96
- segfault_sigaction
 - heap.cpp, 139
 - heap.h, 140
- SHIFT_IMM
 - instruction_impl.h, 158
- SHIFT_REAL
 - instruction_impl.h, 158
- SHImp, 97
 - exec, 98
 - SHImp, 97
- showWarning
 - MainWindow, 77
- SimDef
 - instruction_impl.h, 159
- SimImplDef
 - instruction_impl.h, 159
- size
 - Heap, 48
 - Stack, 111
- SLLImp, 98
 - exec, 99
 - SLLImp, 98
- SLLVImp, 99
 - exec, 100
 - SLLVImp, 99
- SLTIImp, 100
 - exec, 101
 - SLTIImp, 100
- SLTImp, 101
 - exec, 102
 - SLTImp, 101
- SLTIUImp, 102
 - exec, 103
 - SLTIUImp, 102
- SLTUImp, 103
 - exec, 104
 - SLTUImp, 103
- SRAImp, 104
 - exec, 105
 - SRAImp, 104
- SRAVImp, 105
 - exec, 106
 - SRAVImp, 105
- src/executor.cpp, 135
- src/executor.h, 135
- src/fs.h, 135
- src/global.cpp, 136
- src/global.h, 136
- src/heap.cpp, 138
- src/heap.h, 139
- src/instruction.cpp, 140
- src/instruction.h, 141
- src/instruction_impl.cpp, 154
- src/instruction_impl.h, 154
- src/main.cpp, 160
- src/mainwindow.cpp, 161
- src/mainwindow.h, 161
- src/mainwindow.ipp, 164
- src/stack.cpp, 164
- src/stack.h, 164
- src/syscall.h, 166
- SRLImp, 106
 - exec, 107
 - SRLImp, 106
- SRLVImp, 107
 - exec, 108
 - SRLVImp, 107
- STACK
 - mainwindow.h, 163
- Stack, 108
 - ~Stack, 109
 - capacity, 111
 - clear, 109
 - current, 112
 - decrease, 109
 - enlarge, 109
 - get, 110
 - grow, 110
 - highest, 112
 - isEnoughFor, 110
 - order, 111
 - size, 111
 - Stack, 109
- stack
 - MainWindow, 81
- stack.cpp
 - nextPowerOfTwo, 164
- stack.h
 - DEFAULT_SIZE, 165
 - nextPowerOfTwo, 165
 - STACK_HIGH, 165
- STACK_HIGH
 - stack.h, 165
- STATIC
 - mainwindow.h, 163
- STATIC_HIGH
 - global.cpp, 136
 - global.h, 138

STATIC_LOW
 global.h, 137
 SUBImpl, 112
 exec, 113
 SUBImpl, 112
 SUBUImpl, 113
 exec, 114
 SUBUImpl, 113
 SWCLImpl, 114
 exec, 115
 SWCLImpl, 114
 SWImpl, 115
 exec, 116
 SWImpl, 115
 SWLImpl, 116
 exec, 117
 SWLImpl, 116
 SWRImpl, 117
 exec, 118
 SWRImpl, 117
 syscall.h
 SYSCALL_CLOSE, 166
 SYSCALL_EXIT, 166
 SYSCALL_EXIT2, 166
 SYSCALL_FAST_COPY, 166
 SYSCALL_MMAP, 166
 SYSCALL_MUNMAP, 167
 SYSCALL_OPEN, 167
 SYSCALL_PRINT_CHAR, 167
 SYSCALL_PRINT_INT, 167
 SYSCALL_PRINT_STRING, 167
 SYSCALL_READ, 167
 SYSCALL_READ_CHAR, 167
 SYSCALL_READ_INT, 167
 SYSCALL_READ_STRING, 168
 SYSCALL_UI_OPEN_FILE, 168
 SYSCALL_WRITE, 168
 SYSCALL_CLOSE
 syscall.h, 166
 SYSCALL_EXIT
 syscall.h, 166
 SYSCALL_EXIT2
 syscall.h, 166
 SYSCALL_FAST_COPY
 syscall.h, 166
 SYSCALL_MMAP
 syscall.h, 166
 SYSCALL_MUNMAP
 syscall.h, 167
 SYSCALL_OPEN
 syscall.h, 167
 SYSCALL_PRINT_CHAR
 syscall.h, 167
 SYSCALL_PRINT_INT
 syscall.h, 167
 SYSCALL_PRINT_STRING
 syscall.h, 167
 SYSCALL_READ
 syscall.h, 167
 SYSCALL_READ_CHAR
 syscall.h, 167
 SYSCALL_READ_CHAR
 syscall.h, 167
 SYSCALL_READ_INT
 syscall.h, 167
 SYSCALL_READ_STRING
 syscall.h, 168
 SYSCALL_UI_OPEN_FILE
 syscall.h, 168
 SYSCALL_WRITE
 syscall.h, 168
 SYSCALLImpl, 118
 exec, 119
 SYSCALLImpl, 118
 TEQImpl, 119
 exec, 120
 TEQImpl, 119
 TEQImpl, 120
 exec, 121
 TEQImpl, 120
 TGEImpl, 121
 exec, 122
 TGEImpl, 121
 TGEImpl, 122
 exec, 123
 TGEImpl, 122
 TGEIImpl, 123
 exec, 124
 TGEIImpl, 123
 TGEUImpl, 124
 exec, 125
 TGEUImpl, 124
 timer
 MainWindow, 81
 TLTImpl, 125
 exec, 126
 TLTImpl, 125
 TLTImpl, 126
 exec, 127
 TLTImpl, 126
 TLTUImpl, 127
 exec, 128
 TLTUImpl, 127
 TLTUImpl, 128
 exec, 129
 TLTUImpl, 128
 TNEImpl, 129
 exec, 130
 TNEImpl, 129
 TNEImpl, 130
 exec, 131
 TNEImpl, 130
 translateAll
 MainWindow, 77
 TRAP_R
 instruction_impl.h, 159
 TRAP_RI
 instruction_impl.h, 160

- TreeSet
 - heap.h, [140](#)
- TYPE
 - instruction.h, [153](#)
- Ui, [19](#)
- ui
 - MainWindow, [81](#)
- UNLIKELY
 - global.h, [138](#)
- updateAcc
 - MainWindow, [77](#)
- updateHigh
 - MainWindow, [78](#)
- updateLow
 - MainWindow, [78](#)
- updateProgramCounter
 - MainWindow, [78](#)
- updateRegValue
 - MainWindow, [78](#)
- updateStack
 - MainWindow, [79](#)
- XORImpl, [131](#)
 - exec, [132](#)
 - XORImpl, [131](#)
- XORImpl, [132](#)
 - exec, [133](#)
 - XORImpl, [132](#)