Programming Assignment 2

Merritt Vassallo and Naomi Mo

Exercise 1: Original Instructions

```
Cycle  0:  I0:  LI   ar11,   0x9          (VALID)   Commit: None
Cycle  1:  I1:  ADD   ar3,   ar4,  ar5    (VALID)   Commit: I0
Cycle  2:  I2:  LI    ar5,   0x8          (VALID)   Commit: I1
Cycle  3:  I3:  MUL   ar5,   ar5,  ar3    (VALID)   Commit: I2
Cycle  4:                                           Commit: I3
Cycle  5:  I4:  SUB   ar8,  ar11,   ar0   (VALID)   Commit: None
Cycle  6:  I5:  ADD   ar0,   ar0,   ar0   (VALID)   Commit: I4
Cycle  7:  I6:  MUL   ar3,   ar8,  ar25   (INVALID) Commit: I5
Cycle  8:  I7:  ADD   ar7,   ar3,   ar6   (VALID)   Commit: None
Cycle  9:  I8:  SUB   ar7,   ar7,   ar0   (VALID)   Commit: I7
Cycle 10:                                           Commit: I8
```

Instructions After Register Renaming:

I0: LI pr1, 0x9
I1: ADD pr2, pr0, pr0
I2: LI pr3, 0x8
I3: MUL pr4, pr3, pr2
I4: SUB pr3, pr1, pr0
I5: ADD pr0, pr0, pr0
I6: Invalid Instruction
I7: ADD pr5, pr2, pr0
I8: SUB pr6, pr5, pr0

Explanations:

Cycle 0: There are no source registers to rename. Since ar11 appears as the destination register in a valid instruction, we rename ar11 to the lowest-numbered free physical register, pr1. Note that pr0 is not included in the free list upon processor reset; the mapping from ar0 to pr0 is established at reset and pr0 is never deallocated.
ALLOC = {pr0-pr1}, FREE = {pr2-pr63}, MAP = {ar0:pr0, ar11:pr1}, DEALLOC = {}

Cycle 1: Since ar4 appearing as rs1 has not previously been renamed to an architectural register, rs1 here is invalid and should be renamed to pr0 for this instruction. Since ar5 appearing as rs2 has not previously been renamed to an architectural register, rs2 here is invalid and should be renamed to pr0 for this instruction. Since ar3 appears as the destination register in a valid instruction, we rename ar3 to the lowest-numbered free physical register, pr2. Note that although I0 commits in this clock cycle, pr1 does not get deallocated until the

next valid instruction which writes the same architectural register ar11 commits. Because there is no other instruction which writes ar11 in this sequence, pr1 will never get deallocated in this sequence.
ALLOC = {pr0-pr2}, FREE = {pr3-pr63}, MAP = {ar0:pr0, ar11:pr1, ar3:pr2}, DEALLOC = {}

Cycle 2: There are no source registers to rename. Since ar5 appears as the destination register in a valid instruction, we rename ar5 to the lowest-numbered free physical register, pr3. Note that although I1 commits in this clock cycle, pr2 does not get deallocated until the next valid instruction which writes the same architectural register ar3 commits. However, the only other instruction that writes ar3 is I6, which is an invalid instruction, so because there is no other valid instruction which writes ar3 in this sequence, pr2 will never get deallocated in this sequence.
ALLOC = {pr0-pr3}, FREE = {pr4-pr63}, MAP = {ar0:pr0, ar11:pr1, ar3:pr2, ar5:pr3}, DEALLOC = {}

Cycle 3: ar5 appearing as rs1 is renamed to pr3. ar3 appearing as rs2 is renamed to pr2. Since ar5 appears as the destination register in a valid instruction, we rename ar5 to the lowest-numbered free physical register, pr4. Because pr4 replaces pr3 as the physical register to which ar5 is mapped, we take note that the deallocation condition of pr3 is tied to I3 committing. Note that although I2 commits in this clock cycle, pr3 does not get deallocated until the next valid instruction which writes the same architectural register ar5 (in this case I3) commits.
ALLOC = {pr0-pr4}, FREE = {pr5-pr63}, MAP = {ar0:pr0, ar11:pr1, ar3:pr2, ar5:pr4}, DEALLOC = {I3:pr3}

Cycle 4: Physical register pr3 is deallocated upon I3 committing. Note that although I3 commits in this clock cycle, pr4 does not get deallocated until the next valid instruction which writes the same architectural register ar5 commits. Because there are no further instructions which write ar5 in this sequence, pr4 will never get deallocated in this sequence.
ALLOC = {pr0-pr2, pr4}, FREE = {pr3, pr5-pr63}, MAP = {ar0:pr0, ar11:pr1, ar3:pr2, ar5:pr4}, DEALLOC = {}

Cycle 5: ar11 appearing as rs1 is renamed to pr1. ar0 appearing as rs2 is renamed to pr0. Since ar8 appears as the destination register in a valid instruction, we rename ar8 to the lowest-numbered free physical register, pr3.
ALLOC = {pr0-pr4}, FREE = {pr5-pr63}, MAP = {ar0:pr0, ar11:pr1, ar3:pr2, ar8:pr3, ar5:pr4}, DEALLOC = {}

Cycle 6: Architectural register ar0 is always mapped to physical register pr0, whose value is hardwired to zero. Note that although I4 commits in this clock cycle, pr3 does not get deallocated until the next valid instruction which writes the same architectural register ar8 commits. Because there is no other instruction which writes ar8 in this sequence, pr3 will never get deallocated in this sequence.
ALLOC = {pr0-pr4}, FREE = {pr5-pr63}, MAP = {ar0:pr0, ar11:pr1, ar3:pr2, ar8:pr3, ar5:pr4}, DEALLOC = {}

Cycle 7: Since I6 is an invalid instruction, no renaming is performed. Note that although I5 commits in this clock cycle, pr0 does not get deallocated, even when the next valid instruction which writes to the same architectural register ar0 commits. This is because in fact pr0 is a special case and is never deallocated for use in renaming, and is always mapped to ar0.

ALLOC = {pr0-pr4}, FREE = {pr5-pr63}, MAP = {ar0:pr0, ar11:pr1, ar3:pr2, ar8:pr3, ar5:pr4}, DEALLOC = {}

Cycle 8: ar3 appearing as rs1 is renamed to pr2. Since ar6 appearing as rs2 has not previously been renamed to an architectural register, rs2 here is invalid and should be renamed to pr0 for this instruction. Since ar7 appears as the destination register in a valid instruction, we rename ar7 to the lowest-numbered free physical register, pr5.
ALLOC = {pr0-pr5}, FREE = {pr6-pr63}, MAP = {ar0:pr0, ar11:pr1, ar3:pr2, ar8:pr3, ar5:pr4, ar7:pr5}, DEALLOC = {}

Cycle 9: ar7 appearing as rs1 is renamed to pr5. ar0 appearing as rs2 is renamed to pr0. Since ar7 appears as the destination register in a valid instruction, we rename ar7 to the lowest-numbered free physical register, pr6. Because pr6 replaces pr5 as the physical register to which ar7 is mapped, we take note that the deallocation condition of pr5 is tied to I8 committing. Note that although I7 commits in this clock cycle, pr5 does not get deallocated until the next valid instruction which writes the same architectural register ar7 (in this case I8) commits.
ALLOC = {pr0-pr6}, FREE = {pr7-pr63}, MAP = {ar0:pr0, ar11:pr1, ar3:pr2, ar8:pr3, ar5:pr4, ar7:pr6}, DEALLOC = {I8:pr5}

Cycle 10: Physical register pr5 is deallocated upon I8 committing. Note that although I8 commits in this clock cycle, pr6 does not get deallocated until the next valid instruction which writes the same architectural register ar7 commits. There are no more instructions left in the sequence, so it stays allocated in this sequence.
ALLOC = {pr0-pr4, pr6}, FREE = {pr5, pr7-pr63}, MAP = {ar0:pr0, ar11:pr1, ar3:pr2, ar8:pr3, ar5:pr4, ar7:pr6}, DEALLOC = {}