NC State University

Department of Electrical and Computer Engineering

ECE 566: Spring 2024

Project 2

by

RAGHUL SRINIVASAN

UnityID: rsriniv7

(200483357)

Questions

Q1. Describe your implementation and any embellishments you made over my description.

Function Declarations:

- The isDead function checks if an instruction is dead, meaning it has no uses.
- The redundant_instruction function compares two instructions to determine if they are the same and can be eliminated.
- The CseDominatorTree function performs CSE within a dominator tree, recursively traversing basic blocks and eliminating redundant instructions.
- The CommonSubexpressionElimination function is the entry point for the optimization pass. It iterates over functions, basic blocks, and instructions to perform various optimizations, including CSE.

Common Subexpression Elimination (CSE):

- **Function Signature**: The function signature indicates that it takes a pointer to a Module as its argument. This allows the function to operate on the entire module, including all functions and basic blocks within the module.

- **Iterating Over Functions**: The function starts by iterating over all functions in the module using a for loop that iterates over the Module's iterator.

- **Iterating Over Basic Blocks**: Within each function, the function iterates over all basic blocks using another for loop that iterates over the function's iterator.

- **Iterating Over Instructions**: Inside the loop for basic blocks, the function iterates over all instructions in each basic block using a for loop that iterates over the basic block's iterator.

- **Dead Instruction Elimination**: Before performing CSE, the function checks if the current instruction is dead (i.e., its result is not used). If an instruction is dead, it is removed from the parent basic block.

- **Instruction Simplification**: The function checks if the current instruction can be simplified using the simplifyInstruction function. If the instruction can be simplified, it is replaced with the simplified version, and a counter for simplified instructions is incremented.

- **CSE Elimination**: The function then checks for common subexpressions within the current basic block. It iterates over the remaining instructions in the block and compares them with the current instruction to identify redundant instructions. If a redundant instruction is found, it is replaced with the current instruction, and the redundant instruction is erased.

- **Dominator Tree Recalculation**: After performing CSE within a basic block, the function recalculates the dominator tree for the function. This step is necessary to ensure that the dominator tree reflects the changes made to the basic block.

- **Redundant Load Elimination**: If the current instruction is a load, the function checks for redundant loads within the same basic block. If a redundant load is found, it is replaced with the loaded value, and a counter for redundant loads is incremented.

- **Redundant Store Elimination**: If the current instruction is a store, the function checks for redundant stores within the same basic block. If a redundant store is found, the store instruction is removed, and a counter for redundant stores is incremented.

- **Optimization Statistics**: Throughout the function, various counters (CSEDead, CSESimplify, CSEElim, CSELdElim, CSEStore2Load, CSEStElim) are incremented to keep track of the optimizations performed.

- **Overall Module Optimization**: The function iterates over all functions and basic blocks in the module, performing CSE and other optimizations as described above.

Q2. Collect data per benchmark that compares the number of instructions, the total number of loads, the total number of stores, the counters you collect in the CSE pass, and the execution time of your pass both with and without the -mem2reg pass running first. Include this data in either a graph or table in your report.

Number of Instructions

Category	M2R_CSE	M2R_NOCSE	NOM2R_CSE(Default)	NOM2R_NOCSE
adpcm	237	249	402	419
arm	373	429	698	782
basicmath	321	348	552	589
bh	1771	2006	2940	3248
bitcount	418	434	605	655
crc32	83	83	142	145
dijkstra	218	233	309	320
em3d	586	687	1125	1223
fft	380	437	687	724
hanoi	51	53	90	96
hello	2	2	4	4
kmp	329	380	505	556
l2lat	53	58	84	94
patricia	605	718	968	1056
qsort	86	100	134	145
sha	363	418	569	654
smatrix	208	235	279	315
sql	99879	110667	162290	174149
susan	6649	7788	11751	12618

Number of Loads

Category	M2R_CSE	M2R_NOCSE	NOM2R_CSE(Default)	NOM2R_NOCSE
adpcm	15	15	111	122
arm	48	49	212	247
basicmath	24	25	150	167
bh	189	198	761	892
bitcount	50	51	139	175
crc32	8	8	34	37
dijkstra	47	49	92	95
em3d	107	117	359	419
fft	36	38	201	216
hanoi	6	6	25	29
hello	(missing)	(missing)	(missing)	(missing)
kmp	53	58	145	173
l2lat	5	8	17	25
patricia	132	137	345	378
qsort	13	13	35	38
sha	40	42	172	211
smatrix	32	39	71	97
sql	15957	16584	52257	59033
susan	1013	1055	3975	4573

Number of Stores

Category	M2R_CSE	M2R_NOCSE	NOM2R_CSE(Default)	NOM2R_NOCSE
adpcm	7	7	81	81
arm	18	18	116	116
basicmath	12	12	100	100
bh	142	142	494	494
bitcount	17	18	92	98
crc32	4	4	29	29
dijkstra	24	24	51	51
em3d	43	43	192	192
fft	24	24	102	102
hanoi	4	4	16	16
hello	(missing)	(missing)	1	1
kmp	20	20	71	71
l2lat	1	1	14	15
patricia	30	30	108	108
qsort	4	4	16	16
sha	28	28	98	99
smatrix	10	10	31	31
sql	5816	5843	21875	21897
susan	156	157	1429	1438

CSEDead count

Category	M2R_CSE	M2R_NOCSE	NOM2R_CSE(Default)	NOM2R_NOCSE
adpcm	8	(missing)	6	(missing)
arm	40	(missing)	40	(missing)
basicmath	20	(missing)	20	(missing)
bh	78	(missing)	84	(missing)
bitcount	8	(missing)	8	(missing)
crc32	(missing)	(missing)	(missing)	(missing)
dijkstra	8	(missing)	8	(missing)
em3d	39	(missing)	35	(missing)
fft	9	(missing)	8	(missing)
hanoi	1	(missing)	1	(missing)
hello	(missing)	(missing)	(missing)	(missing)
kmp	16	(missing)	16	(missing)
l2lat	1	(missing)	1	(missing)
patricia	47	(missing)	46	(missing)
qsort	4	(missing)	4	(missing)
sha	35	(missing)	36	(missing)
smatrix	1	(missing)	1	(missing)
sql	4203	(missing)	4157	(missing)
susan	34	(missing)	22	(missing)

CSEElim count

Category	M2R_CSE	M2R_NOCSE	NOM2R_CSE(Default)	NOM2R_NOCSE
adpcm	4	(missing)	(missing)	(missing)
arm	15	(missing)	9	(missing)
basicmath	6	(missing)	(missing)	(missing)
bh	148	(missing)	93	(missing)
bitcount	6	(missing)	(missing)	(missing)
crc32	(missing)	(missing)	(missing)	(missing)
dijkstra	5	(missing)	(missing)	(missing)
em3d	52	(missing)	3	(missing)
fft	46	(missing)	14	(missing)
hanoi	1	(missing)	1	(missing)
hello	(missing)	(missing)	(missing)	(missing)
kmp	30	(missing)	7	(missing)
l2lat	1	(missing)	(missing)	(missing)
patricia	61	(missing)	9	(missing)
qsort	10	(missing)	4	(missing)
sha	18	(missing)	9	(missing)
smatrix	19	(missing)	9	(missing)
sql	5931	(missing)	904	(missing)
susan	1062	(missing)	238	(missing)

CSESimplify count

Category	M2R_CSE	M2R_NOCSE	NOM2R_CSE(Default)	NOM2R_NOCSE
adpcm	6	(missing)	5	(missing)
arm	40	(missing)	40	(missing)
basicmath	19	(missing)	19	(missing)
bh	78	(missing)	84	(missing)
bitcount	7	(missing)	7	(missing)
crc32	(missing)	(missing)	(missing)	(missing)
dijkstra	8	(missing)	8	(missing)
em3d	36	(missing)	35	(missing)
fft	9	(missing)	8	(missing)
hanoi	1	(missing)	1	(missing)
hello	(missing)	(missing)	(missing)	(missing)
kmp	16	(missing)	16	(missing)
l2lat	1	(missing)	1	(missing)
patricia	47	(missing)	46	(missing)
qsort	3	(missing)	3	(missing)
sha	35	(missing)	36	(missing)
smatrix	1	(missing)	1	(missing)
sql	4020	(missing)	3974	(missing)
susan	34	(missing)	22	(missing)

CSELdElim count

Category	M2R_CSE	M2R_NOCSE	NOM2R_CSE(Default)	NOM2R_NOCSE
adpcm	(missing)	(missing)	1	(missing)
arm	1	(missing)	28	(missing)
basicmath	(missing)	(missing)	12	(missing)
bh	8	(missing)	77	(missing)
bitcount	(missing)	(missing)	13	(missing)
crc32	(missing)	(missing)	2	(missing)
dijkstra	2	(missing)	3	(missing)
em3d	10	(missing)	18	(missing)
fft	2	(missing)	10	(missing)
hanoi	(missing)	(missing)	4	(missing)
hello	(missing)	(missing)	(missing)	(missing)
kmp	4	(missing)	23	(missing)
l2lat	3	(missing)	6	(missing)
patricia	5	(missing)	27	(missing)
qsort	(missing)	(missing)	3	(missing)
sha	1	(missing)	32	(missing)
smatrix	6	(missing)	25	(missing)
sql	406	(missing)	4218	(missing)
susan	40	(missing)	380	(missing)

CSEStore2Load count

Category	M2R_CSE	M2R_NOCSE	NOM2R_CSE(Default)	NOM2R_NOCSE
adpcm	(missing)	(missing)	10	(missing)
arm	(missing)	(missing)	7	(missing)
basicmath	1	(missing)	5	(missing)
bh	1	(missing)	54	(missing)
bitcount	1	(missing)	23	(missing)
crc32	(missing)	(missing)	1	(missing)
dijkstra	(missing)	(missing)	(missing)	(missing)
em3d	(missing)	(missing)	42	(missing)
fft	(missing)	(missing)	5	(missing)
hanoi	(missing)	(missing)	(missing)	(missing)
hello	(missing)	(missing)	(missing)	(missing)
kmp	1	(missing)	5	(missing)
l2lat	(missing)	(missing)	2	(missing)
patricia	(missing)	(missing)	6	(missing)
qsort	(missing)	(missing)	(missing)	(missing)
sha	1	(missing)	7	(missing)
smatrix	1	(missing)	1	(missing)
sql	221	(missing)	2558	(missing)
susan	2	(missing)	218	(missing)

CSEStElim count

Category	M2R_CSE	M2R_NOCSE	NOM2R_CSE	NOM2R_NOCSE
adpcm	(missing)	(missing)	(missing)	(missing)
arm	(missing)	(missing)	(missing)	(missing)
basicmath	(missing)	(missing)	(missing)	(missing)
bh	(missing)	(missing)	(missing)	(missing)
bitcount	1	(missing)	6	(missing)
crc32	(missing)	(missing)	(missing)	(missing)
dijkstra	(missing)	(missing)	(missing)	(missing)
em3d	(missing)	(missing)	(missing)	(missing)
fft	(missing)	(missing)	(missing)	(missing)
hanoi	(missing)	(missing)	(missing)	(missing)
hello	(missing)	(missing)	(missing)	(missing)
kmp	(missing)	(missing)	(missing)	(missing)
l2lat	(missing)	(missing)	1	(missing)
patricia	(missing)	(missing)	(missing)	(missing)
qsort	(missing)	(missing)	(missing)	(missing)
sha	(missing)	(missing)	1	(missing)
smatrix	(missing)	(missing)	(missing)	(missing)
sql	27	(missing)	22	(missing)
susan	1	(missing)	9	(missing)

Timing.py

Category	.M2R_CSE	.M2R_NOCSE	.NOM2R_CSE(Default)	.NOM2R_NOCSE
adpcm	1.62	1.82	1.47	1.52
arm	0	0	0	0
basicmath	0.03	0.04	0.02	0.03
bh	0.96	0.87	0.73	0.74
bitcount	0.19	0.18	0.11	0.14
crc32	0.15	0.16	0.12	0.17
dijkstra	0.04	0.04	0.04	0.07
em3d	0.29	0.32	0.27	0.29
fft	0.03	0.03	0.02	0.02
hanoi	2.52	2.59	3.48	3.48
kmp	0.15	0.15	0.12	0.15
l2lat	0.04	0.03	0.01	0.03
patricia	0.04	0.04	0.03	0.04
qsort	0.02	0.03	0	0.03
sha	0.04	0.04	0.03	0.04
smatrix	4.05	4.94	4.24	4.65
sql	0	0	0	0
susan	1.06	1.02	0.47	0.45

Q3. Explain the difference in results for these two configurations using your data for Q2.

Number of Instructions:

- M2R reduces the number of instructions in all categories compared to NOM2R (No Memory to Register) configurations, indicating improved code optimization and potentially better performance.
- CSE also reduces the number of instructions, but the effect is not as pronounced as with M2R. This is because CSE mainly focuses on eliminating redundant computations rather than optimizing memory access.

Number of Loads and Stores:

- M2R significantly reduces the number of loads and stores in most categories compared to NOM2R configurations, indicating better utilization of registers and reduced memory access.
- CSE also reduces the number of loads and stores, but the effect is not as significant as with M2R. CSE focuses on eliminating redundant computations, which may not always translate to reducing memory accesses.

CSEDead, CSEElim, CSESimplify, CSELdElim, CSEStore2Load, CSEStElim:

- These counters represent various aspects of common subexpression elimination.
- M2R configurations generally have fewer instances of these counters compared to NOM2R configurations, indicating better elimination of redundant computations and improved code optimization.
- CSE configurations, on the other hand, show a mixed effect on these counters. While some counters decrease, indicating successful elimination of redundant computations, others either remain the same or increase, indicating that CSE may not always be effective in reducing these specific types of operations.

Q4. Compare the output counters you collected. What trends do you notice across the applications for CSE_Basic, CSE_Dead, and CSE_Simplify? ECE 566 students should also include CSE_RLoad, CSE_RStore, and CSE_Store2Load in their answers. Please include data to justify your answer.

CSEDead (CSE_Dead):

- In NOM2R_CSE, enabling only CSE may eliminate some redundant computations, leading to a certain number of dead instructions.
- In M2R_CSE, where memory-to-register optimization is also enabled, the number of dead instructions might be higher because common subexpression elimination can eliminate more computations, especially if they involve memory accesses.

CSEElim (CSE_Basic):

- In NOM2R_CSE, enabling only CSE may eliminate a certain number of common subexpressions, but the scope is limited compared to M2R_CSE.
- In M2R_CSE, common subexpression elimination can be more effective due to the additional optimization of reducing memory accesses, potentially leading to a higher number of eliminated subexpressions.

CSESimplify (CSE_Simplify):

- This metric indicates the number of instructions that are simplified due to CSE.
- Similar to CSEElim, in NOM2R_CSE, the scope of simplification is limited compared to M2R_CSE, where memory-to-register optimization can enable more aggressive simplifications.

CSELdElim (CSE_RLoad):

- This metric indicates the number of loads that are eliminated due to CSE.
- In NOM2R_CSE, the number of loads eliminated may be fewer compared to M2R_CSE, where memory-to-register optimization can enable more loads to be eliminated.

CSEStore2Load (CSE_ Store2Load):

- This metric indicates the number of store-to-load forwarding opportunities that are created due to CSE.
- The impact of this metric may be similar between NOM2R_CSE and M2R_CSE, as both configurations focus on eliminating redundant computations rather than specifically optimizing memory accesses.

CSEStElim (CSE_ RStore):

- This metric indicates the number of stores that are eliminated due to CSE.
- Similar to CSELdElim, the number of stores eliminated may be fewer in NOM2R_CSE compared to M2R_CSE, where memory-to-register optimization can enable more store eliminations.