ECE 566 Project 2

Ibrahim Moghul

1. There are 5 different optimizations that I implemented. The first one removes any dead instructions, which means it has no uses and won't cause any side effects. The second one is to simplify any possible instructions. This will convert a more complex instruction like a multiply for example and convert it into a less complex instruction like a shift or add. The third one is common subexpression elimination. This will look for 2 instructions that are the same and get rid of the redundancy. The fourth and fifth optimization will get rid of redundant loads and stores by checking addresses and operands of future loads and stores and ensuring there is no redundancy.
2.

Instructions			
	Optimized	M2RCSE	NOCSE
adpcm	402	236	419
arm	696	367	782
basicmath	548	312	589
bh	2889	1761	3248
bitcount	603	416	655
crc32	141	83	145
dijkstra	309	216	320
em3d	1121	578	1223
fft	686	376	724
hanoi	90	50	96
hello	4	2	4
kmp	502	322	556
l2lat	80	53	94
patricia	970	594	1056
qsort	134	86	145
sha	569	358	654
smatrix	280	205	315
sql	162178	99192	174149
susan	11743	6482	12618

Loads			
	Optimized	M2RCSE	NOCSE

adpcm	111	15	122
arm	209	46	247
basicmath	148	19	167
bh	762	187	892
bitcount	137	49	175
crc32	33	8	37
dijkstra	92	47	95
em3d	356	102	419
fft	201	34	216
hanoi	25	6	29
hello	0	0	0
kmp	141	46	173
l2lat	17	5	25
patricia	348	125	378
qsort	35	13	38
sha	172	40	211
smatrix	72	32	97
sql	52132	15828	59033
susan	3972	1012	4573

Store			
	Optimized	M2RCSE	NOCSE
adpcm	81	7	81
arm	116	18	116
basicmath	100	9	100
bh	494	142	494
bitcount	92	17	98
crc32	29	4	29
dijkstra	51	24	51
em3d	192	43	192
fft	102	24	102
hanoi	16	4	16
hello	1	0	1
kmp	71	20	71
l2lat	14	1	15
patricia	108	30	108
qsort	16	4	16
sha	98	28	99
smatrix	31	10	31
sql	21875	5814	21897

susan	1429	156	1438

CSEDead			
	Optimized	M2RCSE	NOCSE
adpcm	1	2	0
arm	0	0	0
basicmath	2	1	0
bh	51	1	0
bitcount	1	1	0
crc32	0	0	0
dijkstra	0	0	0
em3d	1	7	0
fft	1	0	0
hanoi	0	0	0
hello	0	0	0
kmp	0	0	0
l2lat	4	0	0
patricia	1	0	0
qsort	1	1	0
sha	0	0	0
smatrix	2	0	0
sql	332	251	0
susan	5	0	0

CSESimplify			
	Optimized	M2RCSE	NOCSE
adpcm	5	6	0
arm	40	42	0
basicmath	19	19	0
bh	84	84	0
bitcount	7	7	0
crc32	0	0	0
dijkstra	8	8	0
em3d	35	36	0
fft	8	9	0
hanoi	1	1	0
hello	0	0	0
kmp	16	16	0
l2lat	1	1	0
patricia	46	48	0

qsort	3	3	0
sha	36	36	0
smatrix	1	1	0
sql	3975	4094	0
susan	22	34	0

CSEElim			
	Optimized	M2RCSE	NOCSE
adpcm	0	5	0
arm	8	17	0
basicmath	1	7	0
bh	94	150	0
bitcount	0	7	0
crc32	0	0	0
dijkstra	0	7	0
em3d	3	55	0
fft	14	48	0
hanoi	1	2	0
hello	0	0	0
kmp	6	30	0
l2lat	0	1	0
patricia	9	64	0
qsort	4	10	0
sha	9	22	0
smatrix	7	22	0
sql	864	6352	0
susan	238	1228	0

CSELdElim			
	Optimized	M2RCSE	NOCSE
adpcm	1	0	0
arm	31	2	0
basicmath	15	2	0
bh	77	10	0
bitcount	20	0	0
crc32	3	0	0
dijkstra	3	2	0
em3d	21	10	0

fft	10	4	0
hanoi	4	0	0
hello	0	0	0
kmp	27	11	0
l2lat	6	3	0
patricia	24	8	0
qsort	3	0	0
sha	32	1	0
smatrix	24	6	0
sql	4209	485	0
susan	384	41	0

CSEStore2Load			
	Optimized	M2RCSE	NOCSE
adpcm	10	0	0
arm	7	1	0
basicmath	4	4	0
bh	53	0	0
bitcount	18	2	0
crc32	1	0	0
dijkstra	0	0	0
em3d	42	1	0
fft	5	0	0
hanoi	0	0	0
hello	0	0	0
kmp	5	1	0
l2lat	2	0	0
patricia	6	4	0
qsort	0	0	0
sha	7	1	0
smatrix	1	1	0
sql	2569	264	0
susan	217	2	0

CSEStElim			
	Optimized	M2RCSE	NOCSE
adpcm	0	0	0
arm	0	0	0
basicmath	0	3	0

bh	0	0	0
bitcount	6	1	0
crc32	0	0	0
dijkstra	0	0	0
em3d	0	0	0
fft	0	0	0
hanoi	0	0	0
hello	0	0	0
kmp	0	0	0
l2lat	1	0	0
patricia	0	0	0
qsort	0	0	0
sha	1	0	0
smatrix	0	0	0
sql	22	29	0
susan	9	1	0

Timing			
	Optimized	M2RCSE	NOCSE
adpcm	1.78	32.86	1.83
arm	0	0	0
basicmath	0.05	0.07	0.04
bh	1.54	2.07	1.7
bitcount	0.28	0.43	0.32
crc32	0.17	0.22	0.17
dijkstra	0.07	0.09	0.08
em3d	0.98	1.18	1.01
fft	0.07	0.07	0.07
hanoi	3.29	0.01	3.37
kmp	0.25	0.3	0.3
l2lat	0.05	0.08	0.04
patricia	0.11	0.1	0.1
qsort	0.06	0.08	0.05
sha	0.05	0.07	0.04
smatrix	4.54	5.66	5.44
sql	0	0	0
susan	1.09	1.97	1.32

3 & 4:

The difference is shown in Optimized vs M2RCSE. The first point of interest is the drastic decrease in loads and stores when using m2r. This also means that the number of redundant loads and stores to get rid of are less. CSE, though needs to remove more instructions with m2r. Simplify and Dead are pretty comparable, but the increase in CSE is notable for m2r over my optimization. For example sql is 6352 vs 864. This is likely due to the fact that the loads and stores are reduced, so there are more instructions that are exposed to each other and aren’t cushioned by memory operations leading to more common subexpressions. The load and store elimination is also less for m2r as there are less of them as aforementioned.