Conversion of Approximation to a Fork-Join Executable

Each fork-join executable has a total of eight cores. One core is dedicated to synchnroizng the cores while all other cores contain a schedule. Each executable is compiled with a list of objects that represent benchmarks. In addition to these list of objects, a main file is provided that contains two arrays of function pointers for each core. Each associated array is titled `fjnodes` and `pjnodes`. To convert an approximation schedule to a fork join executable, each `fjnodes` and `pjnodes` should contain a list of function pointers that comprises the schedule containing the benchmarks. Each benchmark is represented by an object # found below:

Object Representations of Benchmark Tasks

benchmark	object #
bs	object01
bsort100	object02
crc	object03
expint	object04
fft	object05
insertsort	object06
jfdctint	object07
lcdnum	object08
matmult	object09
minver	object10
ns	object11
nsichneu	object12
qurt	object13
select	object14
simple	object15
sqrt	object16
statemate	object17
ud	object18

Conversion of FJ-791

To help understand with how approximation schedules are converted to a fork-join executable, task FJ-791 will be converted. The benchmark schedule of FJ-791 2 Gram can be found below:

core	fj	p	fj
0	cntrl	cntrl	cntrl
1	statemate	select [x 10] statemate [x 7]	select
2	-	select [x 10] statemate [x 7]	-
3	-	select [x 10] statemate [x 7]	-
4	-	select [x 10] statemate [x 6]	-
5	-	select [x 10] statemate [x 6]	-
6	-	select [x 10] statemate [x 6]	-
7	-	select [x 10] statemate [x 6]	-

The next step is to look at each respective benchmark and match it to the object number. Doing so will result in the table found below:

core	fj	p	fj
0	cntrl	cntrl	cntrl
1	17	14 14 14 14 14 14 14 14 14 14 17 17 17 17 17 17 17	14
2	-	14 14 14 14 14 14 14 14 14 14 17 17 17 17 17 17 17	-
3	-	14 14 14 14 14 14 14 14 14 14 17 17 17 17 17 17 17	-
4	-	14 14 14 14 14 14 14 14 14 14 17 17 17 17 17 17	-
5	-	14 14 14 14 14 14 14 14 14 14 17 17 17 17 17 17	-
6	-	14 14 14 14 14 14 14 14 14 14 17 17 17 17 17 17	-
7	-	14 14 14 14 14 14 14 14 14 14 17 17 17 17 17 17	-

Each core is associated with an array titled `fjnode` and `pjnode`. To start the conversion process each core's array should contain all function pointers to each object. An example of core 1 and 2 can be found below:

```
object_t *fjnodes1[NUM_SECTIONS + 1] = {
        object17, // statemate
        object14 // select
};

object_t *pnodes1[NUM_SECTIONS][MAX_SEC_THREADS + 1] = {
    {
        object14, // statemate
        object14, // statemate
        object14, // statemate
        object14, // statemate
```

```
        object14, // statemate
        object14, // statemate
        object14, // statemate
        object14, // statemate
        object14, // statemate
        object14, // statemate
        object17, // select
        object17, // select
        object17, // select
        object17, // select
        object17, // select
        object17, // select
        object17, // select
    }
};

object_t *fjnodes2[NUM_SECTIONS + 1] = {
};

object_t *pnodes2[NUM_SECTIONS][MAX_SEC_THREADS + 1] = {
    {
        object14, // statemate
        object14, // statemate
        object14, // statemate
        object14, // statemate
        object14, // statemate
        object14, // statemate
        object14, // statemate
        object14, // statemate
        object14, // statemate
        object14, // statemate
        object17, // select
        object17, // select
        object17, // select
        object17, // select
        object17, // select
        object17, // select
        object17, // select
    }
};

...
```

After providing all function pointers for each array the conversion of approximation to a fork-join executable is now complete.

List of Approximation Schedules

FJ - 484 Schedule

2 Gram

core	fj	p	fj
0	cntrl	cntrl	cntrl
1	bs	select qurt qurt minver expint simple matmult matmult ud sqrt ns insertsort bsort100 statemate	simple
2	-	select qurt qurt minver expint simple matmult matmult ud sqrt ns insertsort bsort100 statemate	-
3	-	select qurt qurt expint expint simple matmult bs bs bs bs ud ud sqrt insertsort insertsort bsort100 statemate	-
4	-	select qurt qurt expint expint simple matmult bs bs bs bs ud ud sqrt insertsort bsort100 statemate statemate	-
5	-	select qurt qurt expint simple simple matmult bs bs bs bs bs ud sqrt sqrt ns bsort100 statemate statemate	-
6	-	select qurt minver minver minver expint simple matmult matmult ud sqrt sqrt insertsort bsort100 statemate statemate	-
7	-	select qurt minver minver minver expint simple matmult matmult ud sqrt ns insertsort bsort100 statemate	-

3 Parm

core	fj	p	fj
0	cntrl	cntrl	cntrl
1	bs	select [x 7] qurt [x 12] minver [x 8] expint [x 9] simple [x 8] matmult [x 2]	simple
2	-	matmult [x 4]	-
3	-	matmult [x 4]	-
4	-	matmult [x 1] bs(13) ud [x 9] sqrt [x 9] ns [x 4] insertsort [x 7] bsort100 [x 1]	-
5	-	bsort100 [x 5]	-
6	-	bsort100 [x 1] statemate [x 10]	-
7	-		-

3 Parm HD

core	fj	p	fj
0	cntrl	cntrl	cntrl
1	bs	select [x 7] qurt [x 12] minver [x 8] expint [x 9]	simple

		simple [x 8] matmult [x 2]	
2	-	matmult [x 4]	-
3	-	matmult [x 4]	-
4	-	matmult [x 1] bs(13) ud [x 9] sqrt [x 9] ns [x 4] insertsort [x 7] bsort100 [x 1]	-
5	-	bsort100 [x 5]	-
6	-	bsort100 [x 1] statemate [x 10]	-
7	-		-

FJ - 781

2 Gram

core	fj	p	fj
0	cntrl	cntrl	cntrl
1	statemate	select [x 10] statemate [x 7]	select
2	-	select [x 10] statemate [x 7]	-
3	-	select [x 10] statemate [x 7]	-
4	-	select [x 10] statemate [x 6]	-
5	-	select [x 10] statemate [x 6]	-
6	-	select [x 10] statemate [x 6]	-
7	-	select [x 10] statemate [x 6]	-
## 3 Parm			
core	fj	p	fj
----------	----------	----------	----------
0	cntrl	cntrl	cntrl
1	statemate	select [x 70] statemate [x 45]	select
2	-		-
3	-		-
4	-		-
5	-		-
6	-		-
7	-		-
## 3 Parm HD			

core	fj	p	fj
----------	----------	----------	----------
0	cntrl	cntrl	cntrl
1	statemate	select [x 70] statemate [x 45]	select
2	-		-
3	-		-
4	-		-
5	-		-
6	-		-
7	-		-

FJ - 956

2 Gram

core	fj	p	fj
0	cntrl	cntrl	cntrl
1	bsort100	bsort100 [x 10] matmult [x 7]	matmult
2	-	bsort100 [x 10] matmult [x 7]	-
3	-	bsort100 [x 9] matmult [x 8]	-
4	-	bsort100 [x 9] matmult [x 8]	-
5	-	bsort100 [x 9] matmult [x 7]	-
6	-	bsort100 [x 9] matmult [x 7]	-
7	-	bsort100 [x 9] matmult [x 7]	-

3 Parm

core	fj	p	fj
0	cntrl	cntrl	cntrl
1	bsort100	bsort100 [x 20]	matmult
2	-	bsort100 [x 21]	-
3	-	bsort100 [x 21]	-
4	-	bsort100 [x 3] matmult [x 12]	-
5	-	matmult [x 14]	-

6	-	matmult [x 14]	-
7	-	matmult [x 11]	-

3 Parm HD

core	fj	p	fj
0	cntrl	cntrl	cntrl
1	bsort100	bsort100 [x 20]	matmult
2	-	bsort100 [x 20]	-
3	-	bsort100 [x 20]	-
4	-	bsort100 [x 5] matmult [x 10]	-
5	-	matmult [x 14]	-
6	-	matmult [x 14]	-
7	-	matmult [x 13]	-