Compilers Lab Assignemnt 2

Max Marks: 20

1. Write down the CFG for the given language. You are required to give a presentation showcasing your understanding of the grammar for the given language. (10 marks)

2. Construct a lexer for this language. The resulting program shall perform lexical analysis of the given code to generate tokens, to an output file. (6 marks)

3. The remaining 4 marks are reserved for innovative extensions to the language/compiler (eg. process scheduler).

The language

The language is an extension of C language and hence the program for Lexical Analysis should have all necessary provisions to handle cases of arithmetic and logical operation, nested loops (for , while) and nested conditional statements. Here is a brief description for the sample language.

Processor Class

Parameters :

1> isa : ARM,AMD,CDC,MIPS(string)

2> clock_speed : (float)

3> l1_memory : (Memory object name or definition)

4> l2_memory : (Memory object name or definition) or None

5> name : (string) or None

Constructor :

Processor(isa = “ARM”, clock_speed : 40, l1_memory = Mem1)

Member functions:

is_running() : return true or false

submit_jobs(job_id)

submit_jobs([job_id1,job_id2])

get_clock_speed()

run(proc1) , run ([proc1, proc2, ..., procN])

discard_job(job1)

Link Class

Parameters:

1> start_point : (string)

2> end_point : (string)

3> bandwidth : (float)

4> channel_capacity : (float)

5> name : (string) or None

Constructor :

Link(start_point=“proc1”, end_point=“ram”, bandwitdth=55 , 44, “link1”)

Memory Class

Parameters :

1> memory_type :primary, secondary, cache (string)

2> mem_size: (int) in bytes

3> name : (String) or None

Member Functions

get_available_memory()

Constructor :

Memory(memory_type=”primary”, mem_size = 512)

Job Class

Parameters :

1> job_id: (int)

2> flops_required:(int) or (float)

3> deadline:(int) or (float)

4> mem_required (int)

5>affinity(float array)(an array of floats indicating how efficiently the job executes on processors having different instruction set. The order of processors will be AMD,CDC,MIPS,DEC)

Member functions

get_memory()

Constructor :

Job(job_id=1, flops_required= 100, deadline = 50, mem_required = 512,affinity = [0.2,0.5,1,2])

Cluster Class

Parameters:

1> processors : [proc1, proc2, ... , procN] (list of processor objects or List of Clusters)

2> topology : (string) [star, bus, ring etc..]

3> link_bandwidth : (float)

4> link_capacity : (float)

5> name : (string) or None

Constructor:

Cluster(processors=processor_array1,topology = “ring”, 50, 40, name = “cluster1”)

Example for Creating Cluster **:**

```
for(i=0;i<10;i++)
{
        mem1 = Memory(memory_type=”primary”, mem_size = 512 )
        processor_array[i] = Processor(architecture_type=’Embedded_CPU’, isa = “ARM”,
clock_speed : 40, l1_memory = mem1)

}

cluster1 = Cluster(processor_array, “ring”, 50, 40, name = “cluster1”)
```

Following are few **Examples Codes** showing the use of new language :

1.

job_1 = Job(job_id=1, flops_required = 100, deadline = 200, mem_required = 1024, affinity = [0.2,0.5,1,2])

job_2 = Job(job_id=2, flops_required = 5, deadline = 20, mem_required = 64, affinity = [0.2,0.5,1,2])

mem1 = Memory(memory_type= ‘cache’, mem_size=1)

ram = Memory(memory_type= ‘primary’, mem_size = 2048,, name = “ram1”)

proc_1 = Processor(isa = ‘ARM’, clock_speed : 40, l1_memory = mem1)

link_1 = Link(start_point = “proc1”, end_point= “ram1”, 40, 50)

proc_1.submit_jobs([job_1,job_2,job_3])

run(proc_1)

2.

job_1 = Job(job_id=1, flops_required = 100, deadline = 200, mem_required = 1024, affinity = [0.2,0.5,1,2])

job_2 = Job(job_id=2, flops_required = 5, deadline = 20, mem_required = 64, affinity = [0.2,0.5,1,2])

job_3 = Job(job_id=3, flops_required = 5, deadline = 10, mem_required = 64, affinity = [0.2,0.5,1,2])

mem1 = Memory(memory_type= ‘cache’, mem_size=1)

mem2 = Memory(memory_type= ‘cache’, mem_size=0.5)

mem3 = Memory(memory_type= ‘cache’, mem_size=1.5)

ram = Memory(memory_type= ‘primary’, mem_size = 2048,, name = “ram1”)

proc_1 = Processor(isa = ‘ARM’, clock_speed : 40, l1_memory = mem1)

proc_2 = Processor(isa = ‘AMD’, clock_speed : 60, l1_memory = mem2, name = “proc_2”)

proc_3 = Processor(isa = ‘MIPS’, clock_speed : 20, l1_memory = mem3, name = “proc_3”)

link_1 = Link(start_point = “proc_1”, end_point= “ram1”, 40, 50)

link_2 = Link(start_point = “proc_2”, end_point= “ram1”, 40, 50)

link_3 = Link(start_point = “proc_3”, end_point= “ram1”, 40, 50)

proc_1.submit_jobs(job_1)

proc_2.submit_jobs(job_2)

proc_3.submit_jobs(job_3)

run([proc_1,proc_2,proc_3])

3.

job_1 = Job(job_id=1, flops_required = 100, deadline = 200, mem_required = 1024,affinity = [0.2,0.5,1,2])

job_2 = Job(job_id=2, flops_required = 5, deadline = 20, mem_required = 64, affinity = [0.2,0.5,1,2])

mem1 = Memory(memory_type= ‘cache’, mem_size=1)

ram = Memory(memory_type= ‘primary’, mem_size = 2048,, name = “ram1”)

proc_1 = Processor(isa = ‘ARM’, clock_speed : 40, l1_memory = mem1)

link_1 = Link(start_point = “proc_1”, end_point= “ram1”, 40, 50)

```
while(! Ram.get_available_memory())
{
        wait(1)
}

if job_1.get_memory() <= ram.get_available_memory()
{
        proc_1.submit_jobs(job_1)
}
else
{
        discard_job(job_1)
}
```

4.

```
for ( i=0;i<10;i++)
{
	job_array[i] = Job(job_id=i, flops_required = 10, deadline = 10 + i*5 , mem_required = 64, affinity = [0.2,0.7,1,2])
}

for ( i=0;i<10;i++)
{
	mem1 = Memory(memory_type=’primary’, mem_size = 512 )
	processor_array[i] = Processor(isa = ‘ARM’, clock_speed : 40, l1_memory = mem1)
}

ram = Memory(memory_type= ‘primary’, mem_size = 4096,, name = “ram1”)

cluster_1 = Cluster(processors=processor_array,topology =  “ring”, 50, 40, name = “cluster1”)

link_1 = Link(start_point = “cluster1”, end_point= “ram1”, 40, 50)

cluster_1.submit_jobs(job_array)

run(cluster_1)
```

5.

```
for ( i=0;i<10;i++)
{
	job_array[i] = Job(job_id=i, flops_required = 10, deadline = 10 + i*5 , mem_required = 64, affinity = [0.2,0.7,1,2])
}

for ( i=0;i<5;i++)
{
	mem1 = Memory(memory_type=’primary’, mem_size = 512 )
	processor_array1[i] = Processor( isa = ‘ARM’, clock_speed : 40, l1_memory = mem1)

	processor_array2[i] = Processor( isa = ‘AMD’, clock_speed : 80, l1_memory = Memory(memory_type=’primary’, mem_size = 512 ))
}

ram = Memory(memory_type= ‘primary’, mem_size = 4096,, name = “ram1”)

cluster_1 = Cluster(processors=processor_array1,topology =  “ring”, 50, 40, name = “cluster1”)

cluster_2 = Cluster(processors=processor_array2,topology =  “star”, 50, 40, name = “cluster2”)

link_1 = Link(start_point = “cluster1”, end_point= “ram1”, 40, 50)
```

```
link_2 = Link(start_point = “cluster2”, end_point= “ram1”, 40, 50)

cluster_3 = Cluster(processors=[cluster_1, cluster_2],topology =  “star”, 100, 80, name =
“cluster3”)

cluster_3.submit_jobs(job_array1)

run(cluster3)
```