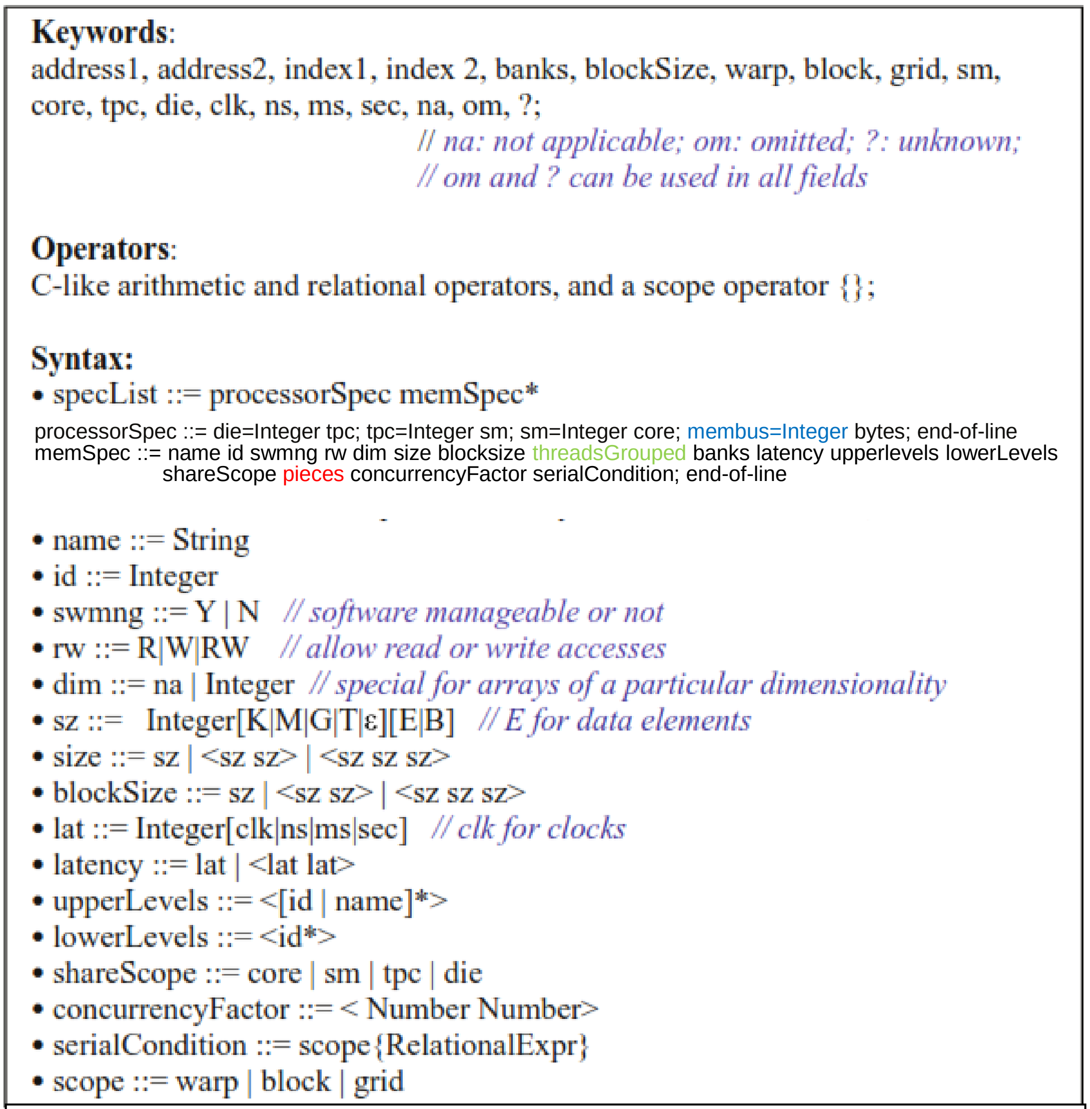
Syntax of MSL

Others are same except for:



Keywords:
address1, address2, index1, index 2, banks, blockSize, warp, block, grid, sm, core, tpc, die, clk, ns, ms, sec, na, om, ?;
// na: not applicable; om: omitted; ?: unknown;
// om and ? can be used in all fields

Operators:
C-like arithmetic and relational operators, and a scope operator {};

Syntax:

- specList ::= processorSpec memSpec*

processorSpec ::= die=Integer tpc; tpc=Integer sm; sm=Integer core; membus=Integer bytes; end-of-line
memSpec ::= name id swmng rw dim size blocksize threadsGrouped banks latency upperlevels lowerLevels
shareScope pieces concurrencyFactor serialCondition; end-of-line

- name ::= String
- id ::= Integer
- swmng ::= Y | N *// software manageable or not*
- rw ::= R|W|RW *// allow read or write accesses*
- dim ::= na | Integer *// special for arrays of a particular dimensionality*
- sz ::= Integer[K|M|G|T|ε][E|B] *// E for data elements*
- size ::= sz | <sz sz> | <sz sz sz>
- blockSize ::= sz | <sz sz> | <sz sz sz>
- lat ::= Integer[clk|ns|ms|sec] *// clk for clocks*
- latency ::= lat | <lat lat>
- upperLevels ::= <[id | name]*>
- lowerLevels ::= <id*>
- shareScope ::= core | sm | tpc | die
- concurrencyFactor ::= < Number Number>
- serialCondition ::= scope{RelationalExpr}
- scope ::= warp | block | grid

threadsGrouped ::= Integer | <Integer|Integer>

pieces ::= Integer

Example: Mem spec of Tesla M2075:

die = 16 tpc; tpc = 1 sm; sm =32 cores; membus = 48 bytes;

globalMem 8 Y rw na 5375M **128B** 32 ? 600clk<L2 L1> <> die 1 <0.1 0.5> warp{address1/blockSize!= address2/blockSize} ;

L1 9 N rw na 16K 128B 32 ? 80clk <> <L2 globalMem> sm 1 ? warp{address1/blockSize!= address2/blockSize};

L2 7 N rw na 768K 32B <32|4> ? 390clk om om die 2 ? warp{ thread1/<32|4>!=thread2/<32|4> || address1/blockSize != address2/blockSize }; *//address1 and address2 are the transformed addresses in L2*

constantMem 1 Y r na 64K ? 32 ? 360clk <cL2 cL1> <> die 1 ? warp{address1 != address2};

cL1 3 N r na 4K 64B 32 ? 48clk <> <cL2 constantMem> sm 1 ? warp{address1/blockSize!= address2/blockSize};

cL2 2 N r na 32K 256B 32 ? 140clk <cL1> **<constantMem>** die 1 ? warp{address1/blockSize!= address2/blockSize};

sharedMem 4 Y rw na 48K ? 32 32 48clk <> <> sm 1 ? block{word1!=word2&&word1%banks ==word2%banks};

tL1 6 N r na 12K **<32B 4>** 4 ? 208clk <> <L2 textureMem> sm 1 ? warp{ thread1/4!=thread2/4 || address1/blockSize.x!= address2/blockSize.x}; *//address1 and address2 are the transformed addresses in tL1*

textureMem 5 Y r na 5375M na 4 ? 617clk <L2 tL1> <> die 1 <0.1 0.5> ?;

textureMem 5 om om 1 128ME **32B** om ? ? om om om om om warp{thread1/4!= thread2/4 || address1/blockSize != address2/blockSize};

textureMem 5 om om 2 <64KE 64KE > **<16B 2>** om ? ? om om om om warp{thread1/4!= thread2/4 || address1.x/blockSize.x!= address2.x/blockSize.x || address1.y/blockSize.y!= address2.y/blockSize.y }