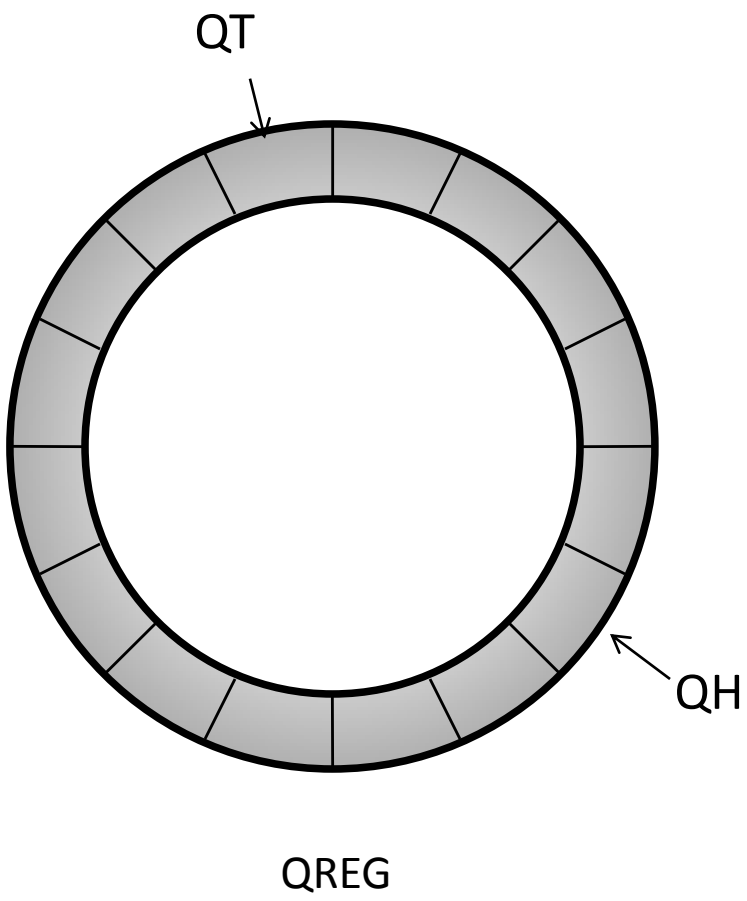
Qasm

Technical Report, Ref. TR12010

ASL-Ben Abdallah Group
The University of Aizu,
Adaptive Systems Laboratory, School of Computer Science and Engineering

Background (1)

- What is Queue Computation?
  - The intermediate data is written into a circular queue register (QREG)
  - A given instruction implicitly reads data from a head of the queue register (QH)
  - The executed result is written into a tail of the queue register (QT).



Background (2)

AIZU
THE UNIVERSITY OF AIZU

- Queue computation features
  - High instruction parallelism
  - Small program size
  - No false data dependency

Background (3)

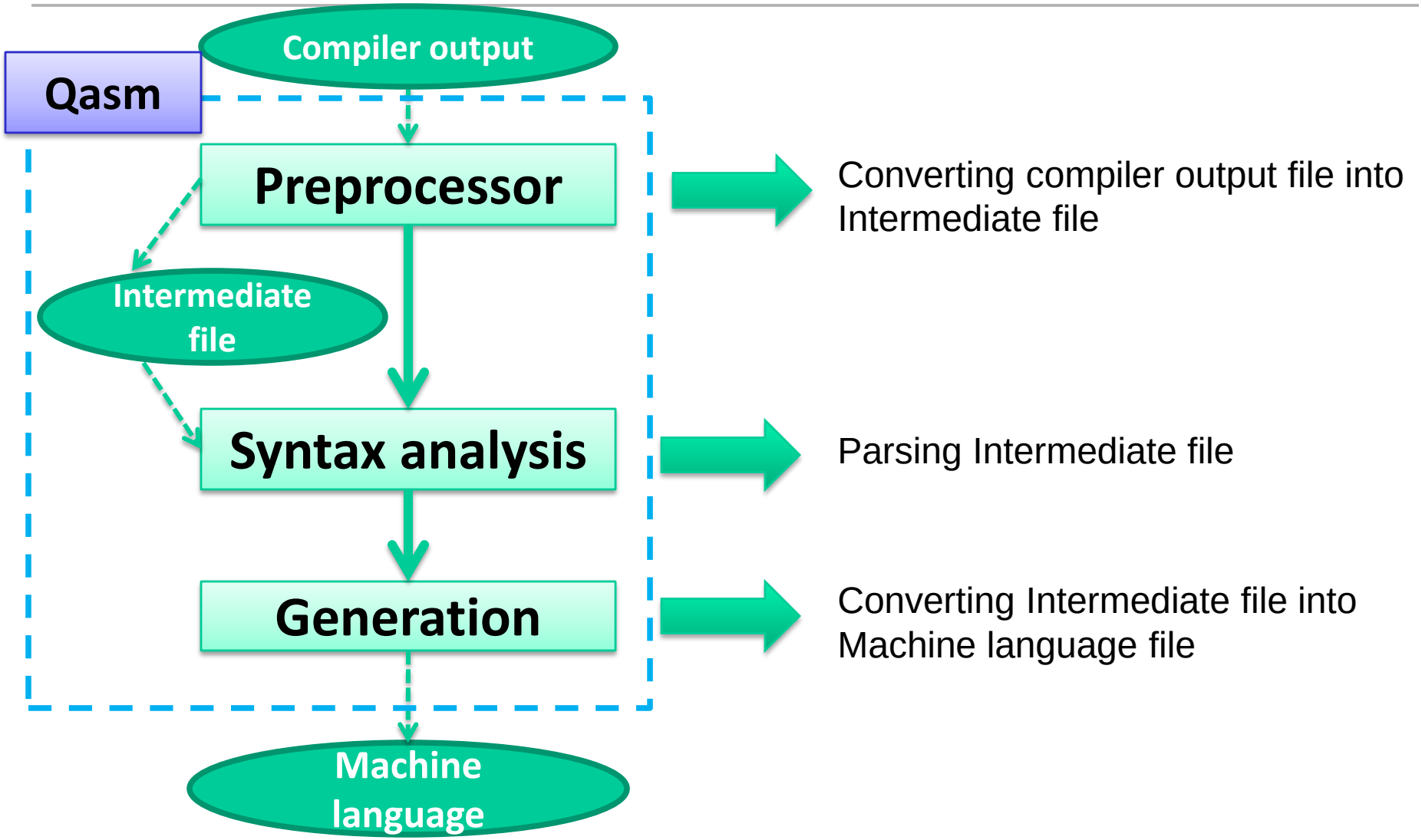
AIZU
THE UNIVERSITY OF AIZU

- In our lab, there are two types of Queue processors
  - Queue Core (QC) model
  - Dual Execution Processor (DEP) model
- Increase the these Queue processors usage by to develop Queue Assembler(Qasm)

Qasm features

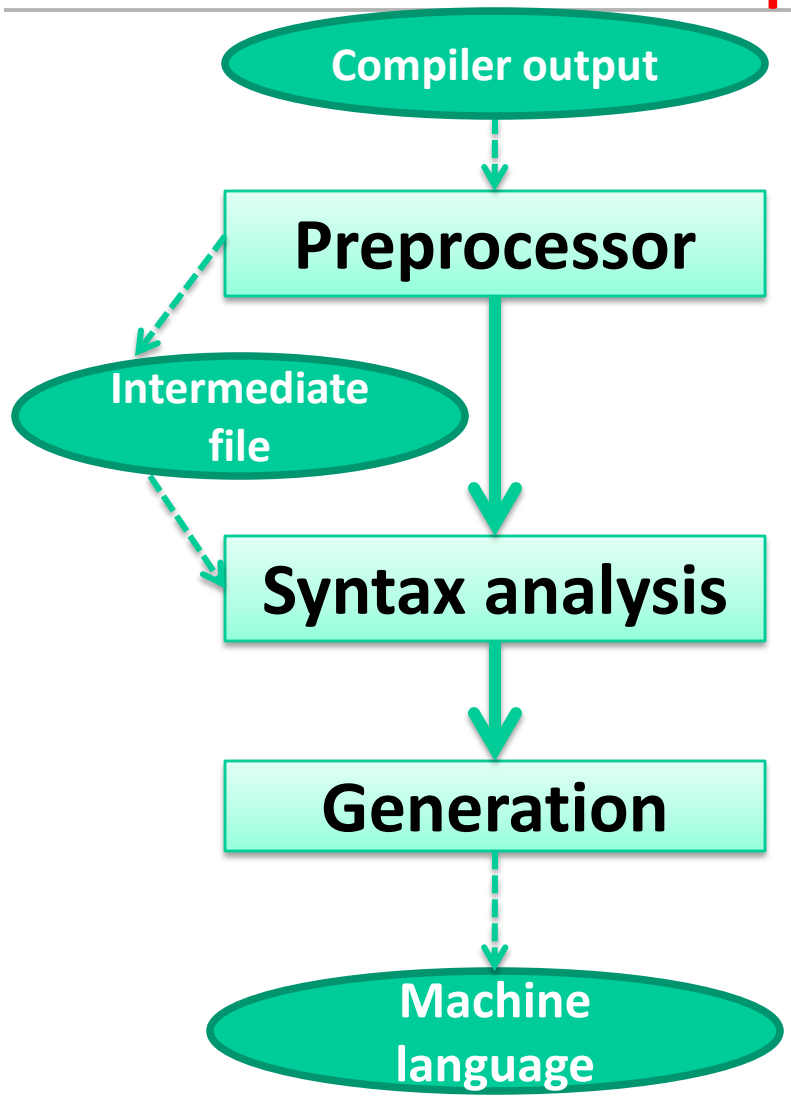
- Development of Queue Assembler(Qasm)
  - User friendly
  - Support two computing models
    - QC and DEP model
  - Support Queue compiler output
    - With preprocessor

Assembler structure



Assembler development

-Preprocessor-



1. Read in one line

sub iws, qt, qh, qh+1

2. Divide into tokens

sub | iws, | qt, | qh, | qh+1

Not needed

3. Clean not needed tokens

sub qh+1

4. Clean not need characters

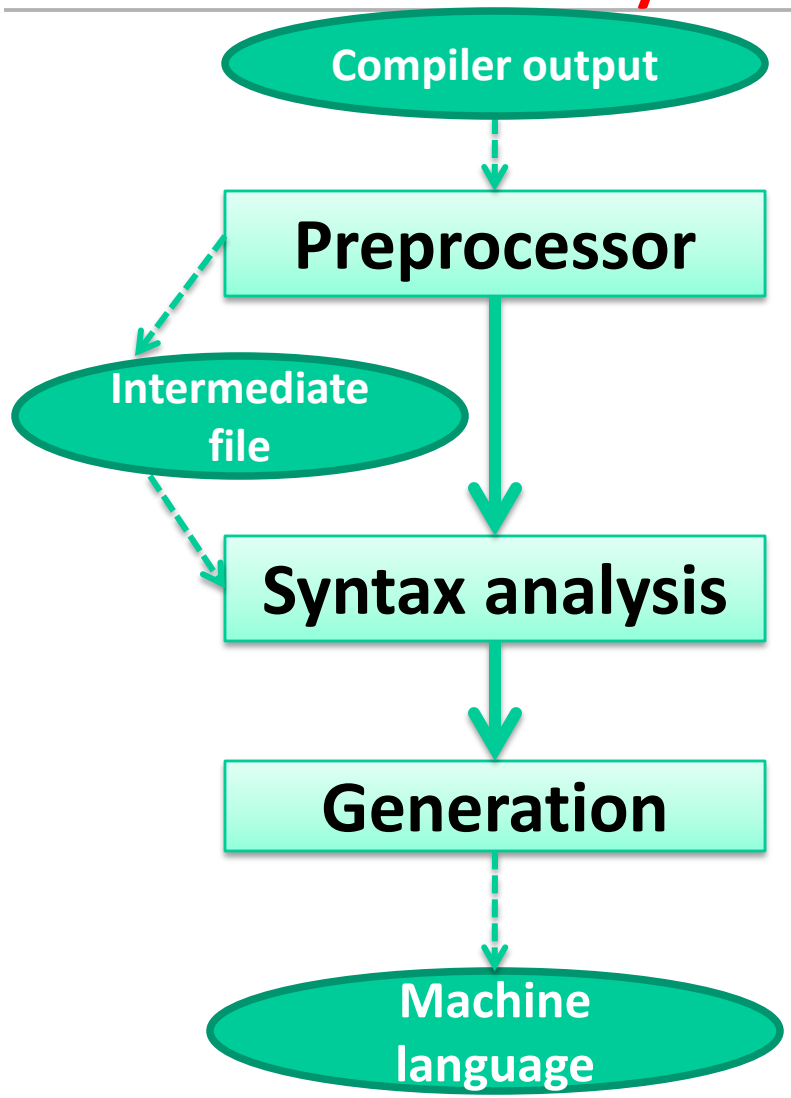
sub 1

5. Combine tokens

sub 1

Assembler development

-Syntax analysis-



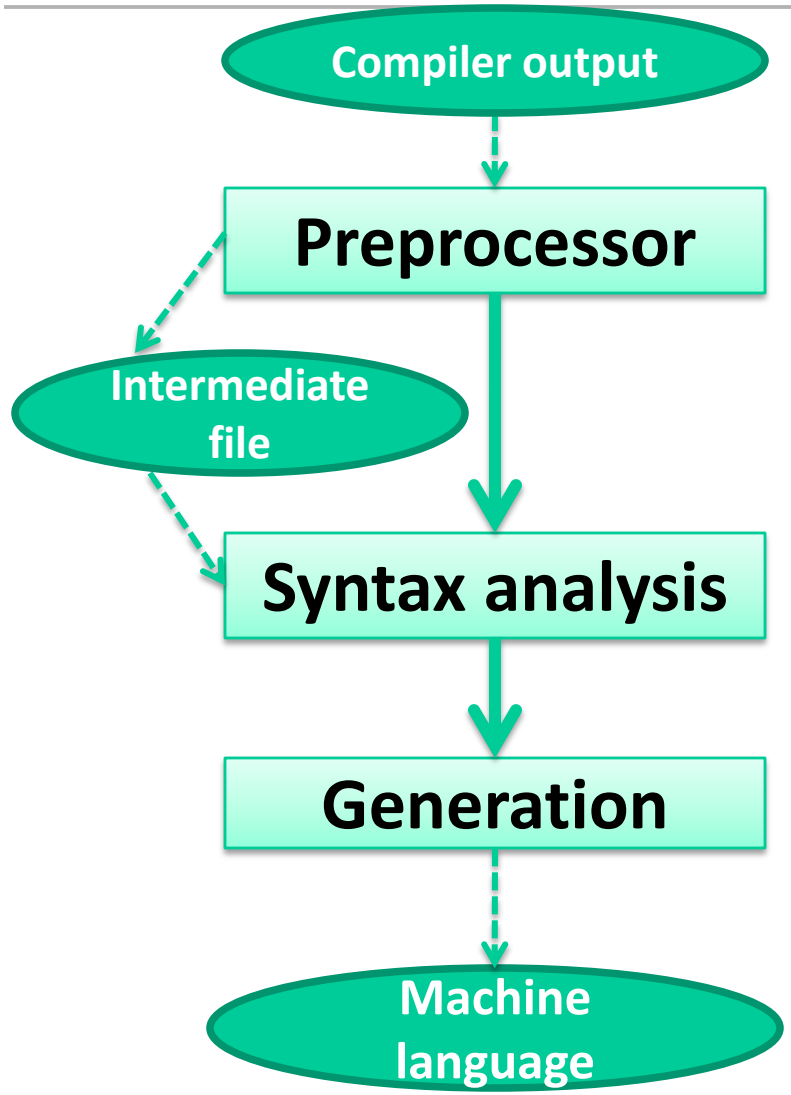
1. Syntax parsing Intermediate file and divide into tokens

main:	add	1	# hoge

2. Divided tokens into category and substitute to structure

main:	add	1	#hoge
Label	Mnemonic	Operand	Coment

Assembler development
-Generation(1)-



1. Replace mnemonic code by opcode

main: add 1 #hoge

MD file

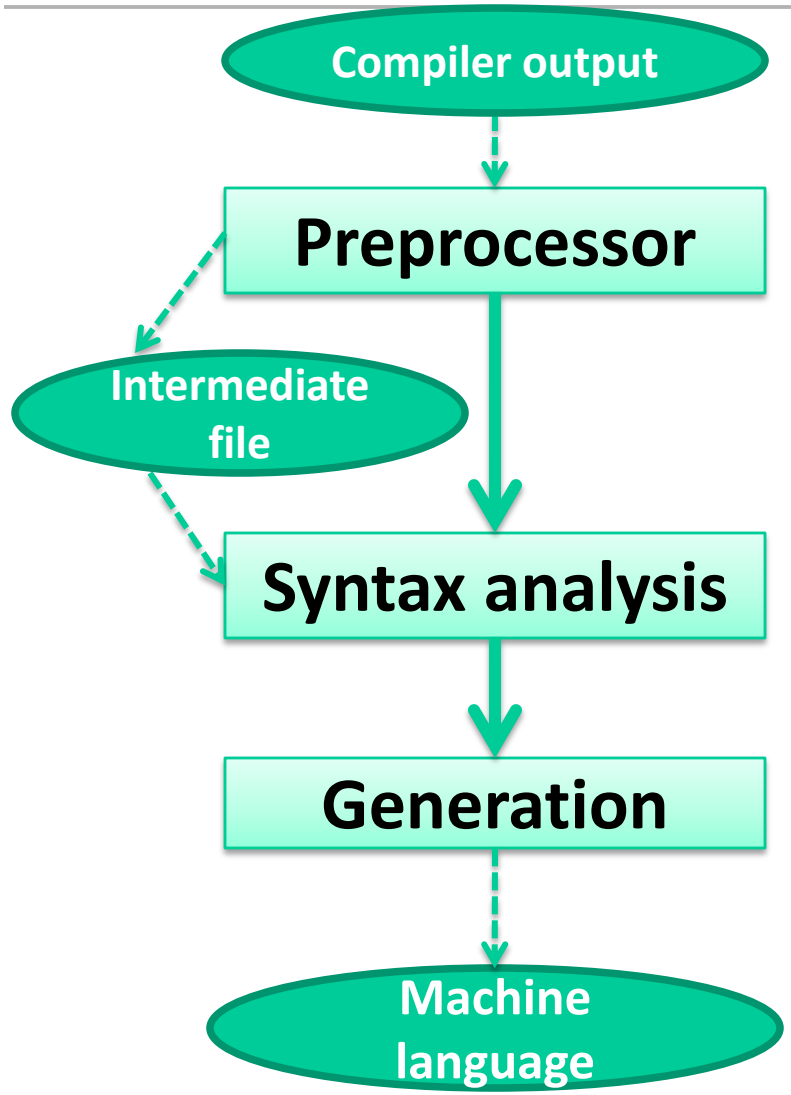
0x30

2. Convert integer into hexadecimal on operand

main: 0x30 1 #hoge

01

Assembler development
-Generation(2)-

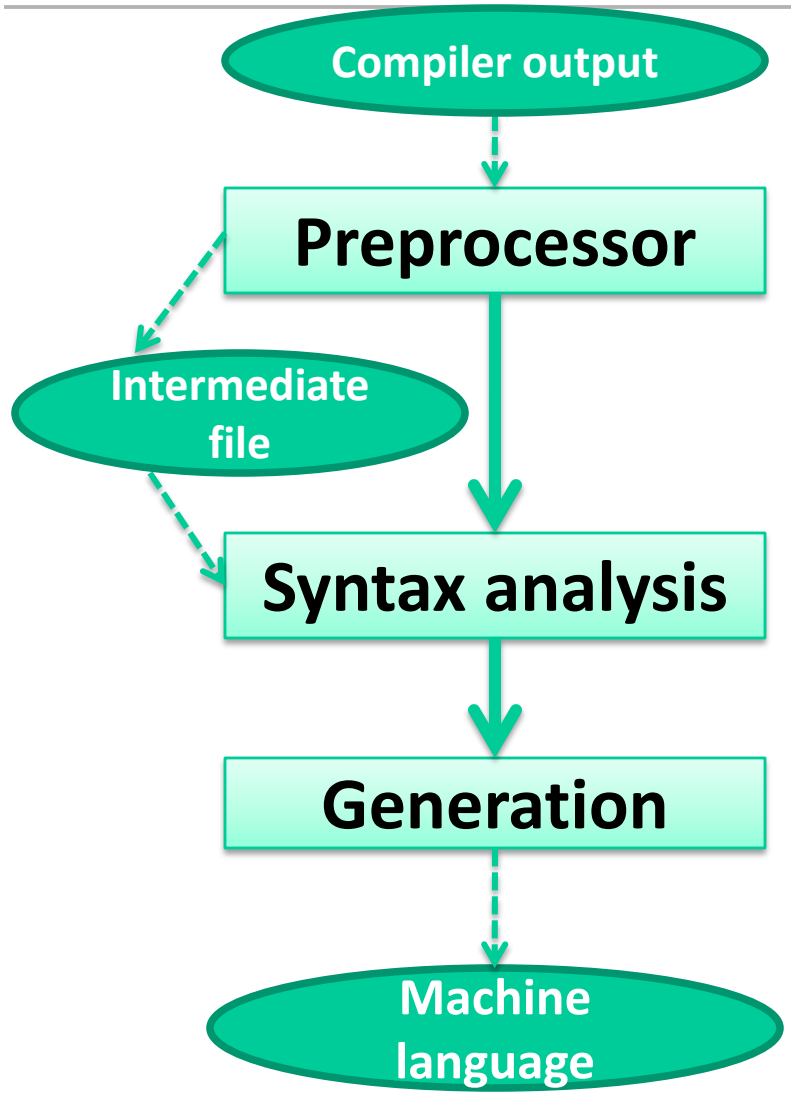


3. Paste address into label

```
.text
        setd0   10      # N
        setd1   2       # counter
        setd2   1       # arg(0)
        setd3   1       # arg(1)
        seta0   $main
        call    0(a0)
        hlt

main:
        ldw     0(d0)
        ldil    2
```

Assembler development
-Generation(2)-



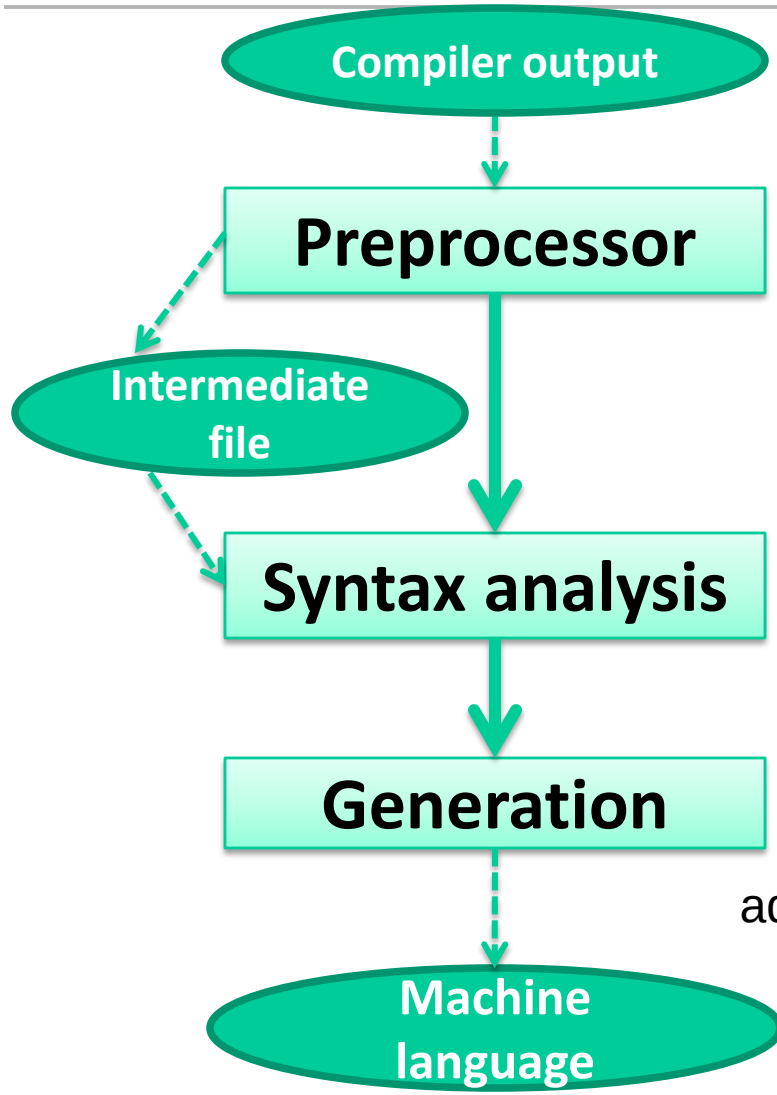
3. Paste address into label

```
.text
        setd0   10      # N
        setd1   2       # counter
        setd2   1       # arg(0)
        setd3   1       # arg(1)
        seta0   $main
        call    0(a0)
        hlt

main:
        ldw     0(d0)
        ldil    2
```

Find a Label

Assembler development
-Generation(2)-



3. Paste address into label

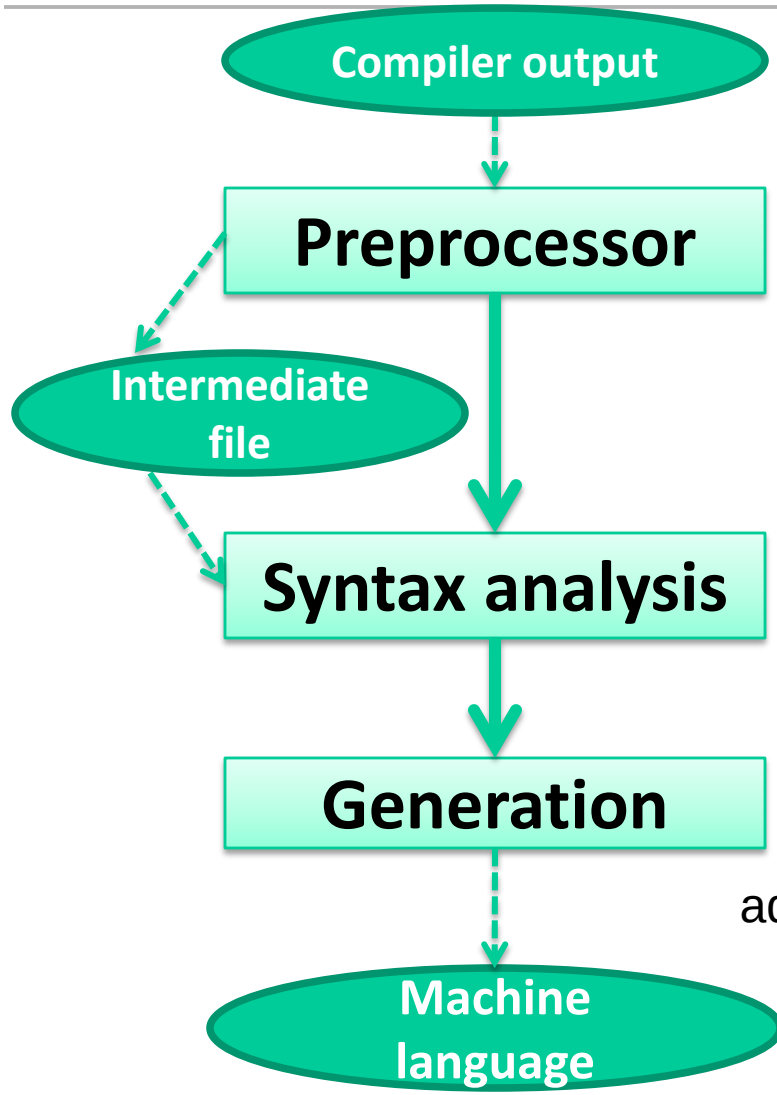
```
.text
        setd0   10      # N
        setd1   2       # counter
        setd2   1       # arg(0)
        setd3   1       # arg(1)
        seta0   $main
        call    0(a0)
        hlt

main:
        ldw     0(d0)
        ldil    2
```

address = 0x0c

Search for jump address

Assembler development
-Generation(2)-



3. Paste address into label

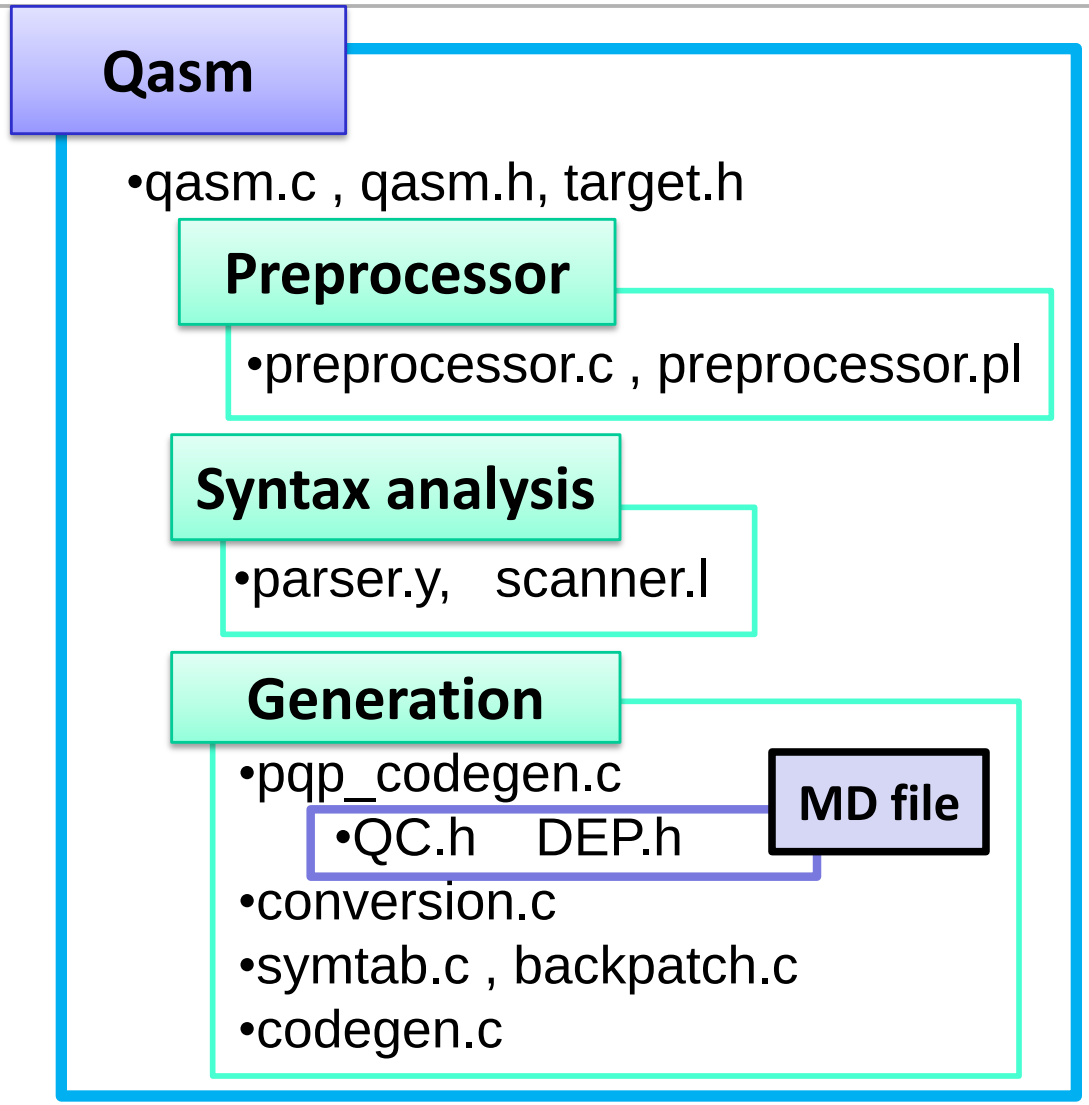
```
.text
        setd0   10      # N
        setd1   2       # counter
        setd2   1       # arg(0)
        setd3   1       # arg(1)
        seta0   $main
        call    0(a0)
        hlt

main:
        ldw     0(d0)
        addil   2
```

address = 0x0c

Change the label to address

Design result(1)



Design result(2)

File	Number of lines	File	Number of lines
qasm.c	400	pqp_codegen.c	579
qasm.h	148	QC.h	420
target.h	37	DEP.h	205
preprocessor.c	62	conversion.c	86
preprocessor.pl	99	symtab.c	332
parser.y	226	backpatch.c	105
scanner.l	348	codegen.c	171

Total number of lines: 3218

Evaluation results(1)

- Qasm executes
  - Reading and writing file assignment
  - Select using compiler output or handmade assembly
  - Decide file type of input and output
  - With user interface

```
std4dc17{s1140196}155: ./qasm
+-----------------------------------------+
|          Queue Computer Project         |
|            University of Aizu           |
|                                         |
|              Queue Assembler            |
|         Optimized by Reo Honjoya,       |
|                        Hiroki Hoshino   |
|          Last update : Feb 04, 2010     |
+-----------------------------------------+
# Do you use compiler output file?
# ( 1:Yes 2:No )
  > 2
# Enter a assembly file name
  > ./Assembly/TEMP/optimize.s
# Enter ISA ( 1:QC model 2:DEP model )
  > 1
# Enter a output file name
  > ./Out/com_out.hex
# Enter output file type ( 1:Binary  2:Hexadecimal )
  > 2
14 bytes written to file: ./Out/com_out.hex
std4dc17{s1140196}156:
```

Evaluation results(2)

```
        .text
main:
        ld      iws,    qt,     ($fp)0
        ld      iws,    qt,     ($fp)4
        sub     iws,    qt,     qh,     qh+1
        mul     iws,    qt,     qh-1,   qh-1
        sub     iws,    qt,     qh,     qh-1
        div     iws,    qt,     qh,     qh+1
        st      iws,    qh,     ($fp)8
```

Queue compiler generated code for (a*a)/{(b-a)-a}

Preprocessor

```
        .text
main:
        ldw     0(d0)
        ldw     0(d1)
        sub     1,  1
        mul     1,  0
        sub     1,  -1
        div     1,  1
        stw     0(d2)
```

Intermediate file

Assemble

```
60 00   // ldw
61 00   // ldw
82 81   // sub
b8 80   // mul
82 ff   // sub

ba 81   // div
7a 00   // stw
```

Machine language