



University of San Carlos | Department of
COMPUTER ENGINEERING

CpE 3202
Computer Organization & Architecture

TRACS Assembler: a briefing

Assembler

- An assembler is a program that takes basic computer instructions and converts them into a pattern of bits that the computer's processor can use to perform its basic operations.
- These instructions are referred to as assembler language or the a more common term **assembly language**.

Assembler

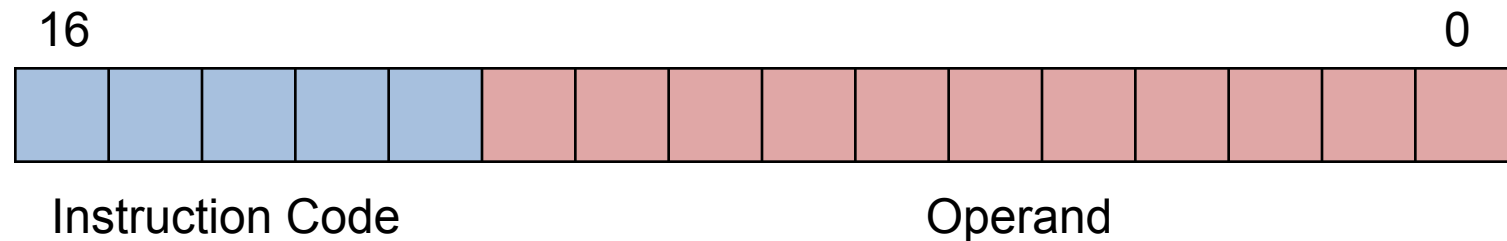
- The assembler also provides program error detection such as syntax, illegal or missing operands, out of range memory address etc. It also process part of the assembly code that is not instruction related also known as "pre-processor directives".

TRACS Assembly Language

- TRACS assembly language is similar to other assembly language from other architecture.
- TRACS instructions are basically with or without operands.
 - **WB 0x08** (instruction with a specified operand)
 - **ADD** (instruction without an operand, implied operation)
- Operands can be an 8-bit data or an 11-bit memory address depending on the instruction.

Converting Assembly Program to Machine Code

- Each instruction has a unique instruction code which is used by the control unit to determine which operation to run during the execute cycle.
- The instruction format for TRACS:



Converting Assembly Program to Machine Code

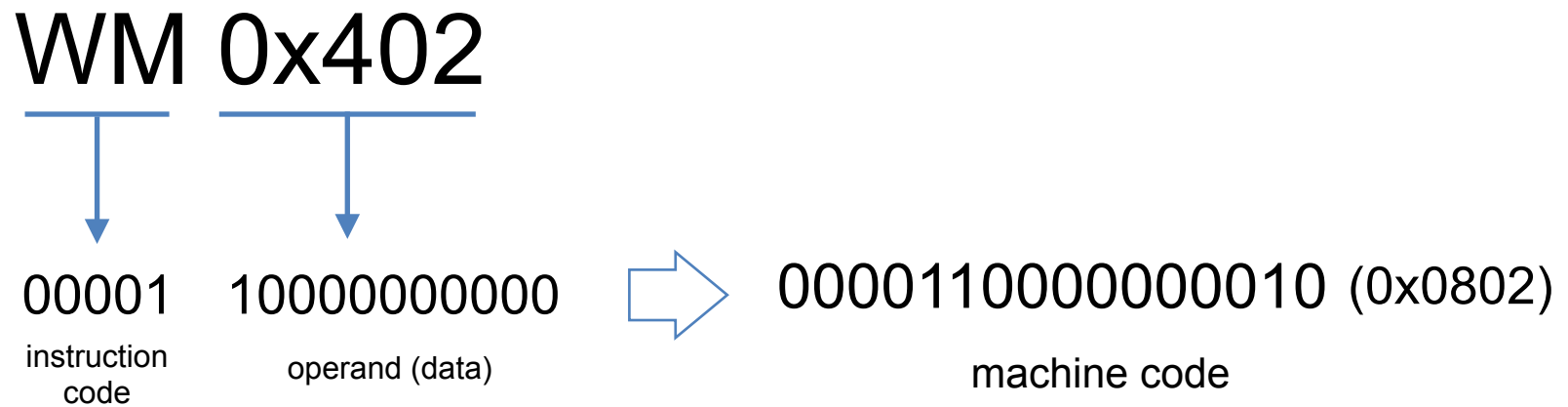
- For example (data operand):



Assume 'x' is '0'.

Converting Assembly Program to Machine Code

- For example (address operand):



- For example (address operand):



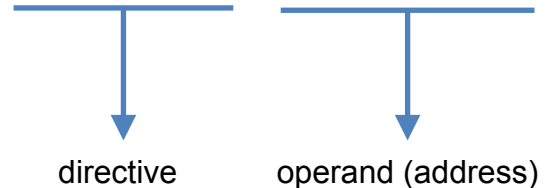
Non-Instruction Directives

- Also known as the “pre-processor” directives.
- These are set of keywords with required syntax to provide information to the processor such as start address of the program or data memory, address of a certain part of the code (labels), comments, constants, files associated to the program and many others.
- These are not executed by the processor but is processed (or not processed in the case of comments) by the assembler.

Non-Instruction Directives

- For example, the ORG directive will signify the start of the program memory. Therefore, the program counter will point to the address specified upon reset.

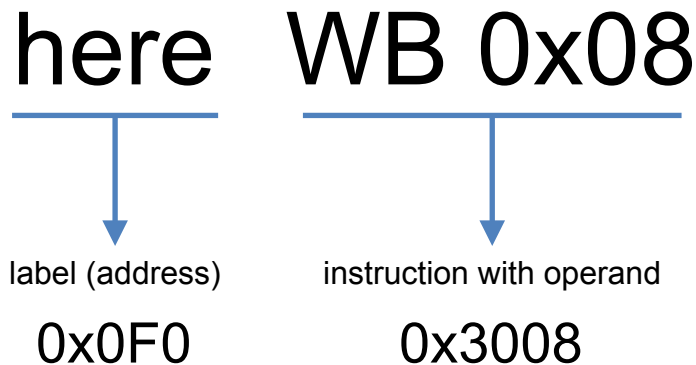
ORG 0x000



directive operand (address)

Non-Instruction Directives

- Another example are “labels”. Labels represents a program memory address so that it would easier for the programmer to jump to a certain part of the code without knowing the actual memory address.



Based on the start address of the program memory, the assembler has to figure out the address represented by the instruction.

TRACS Assembly Program

ORG 0x010

```
start  WB 0x05           ; write 0x05 to MBR
      WM 0x400           ; write data on MBR to memory at address 0x400
      WB 0x03           ; write 0x03 to MBR
      WM 0x401           ; write data on MBR to memory at address 0x401
      RM 0x400           ; read address 0x400, store data to MBR
      WACC               ; write MBR to ACC (via BUS)
      RM 0x401           ; read data at memory address 0x401, store to MBR
      ADD                ; add ACC to MBR (through BUS)
      RACC               ; read ACC, store to MBR
      WM 0x402           ; write data on MBR to memory address 0x402
      WB 0x00           ; write 0x00 to MBR
      RM 0x402           ; read address 0x402, store data to MBR
      WACC               ; write MBR to ACC (via BUS)
      WB 0x05           ; write 0x05 to MBR
      BRGT here         ; compare ACC with MBR, jump to here if greater than
      RACC               ; read ACC, store to MBR
      WM 0x403           ; write data on MBR to memory address 0x403
here SHL                ; shift left ACC
      RACC               ; read ACC, store to MBR
      WM 0x405           ; write data on MBR to memory at address 0x405
      EOP
```

TRACS Assembly Program

ORG 0x010

start

```
WB 0x05
WM 0x400
WB 0x03
WM 0x401
RM 0x400
WACC
RM 0x401
ADD
RACC
WM 0x402
WB 0x00
RM 0x402
WACC
WB 0x05
BRGT here
RACC
WM 0x403
here SHL
RACC
WM 0x405
EOP
```

```
; write 0x05 to MBR
; write data on MBR to memory at address 0x400
; write 0x03 to MBR
; write data on MBR to memory at address 0x401
; read address 0x400, store data to MBR
; write MBR to ACC (via BUS)
; read data at memory address 0x401, store to MBR
; add ACC to MBR (through BUS)
; read ACC, store to MBR
; write data on MBR to memory address 0x402
; write 0x00 to MBR
; read address 0x402, store data to MBR
; write MBR to ACC (via BUS)
; write 0x05 to MBR
; compare ACC with MBR, jump to here if greater than
; read ACC, store to MBR
; write data on MBR to memory address 0x403
; shift left ACC
; read ACC, store to MBR
; write data on MBR to memory at address 0x405
```

pre-processor directives

TRACS Assembly Program

ORG 0x010

start	WB 0x05 WM 0x400 WB 0x03 WM 0x401 RM 0x400 WACC RM 0x401 ADD RACC WM 0x402 WB 0x00 RM 0x402 WACC WB 0x05 BRGT here RACC WM 0x403 here SHL RACC WM 0x405 EOP	; write 0x05 to MBR ; write data on MBR to memory at address 0x400 ; write 0x03 to MBR ; write data on MBR to memory at address 0x401 ; read address 0x400, store data to MBR ; write MBR to ACC (via BUS) ; read data at memory address 0x401, store to MBR ; add ACC to MBR (through BUS) ; read ACC, store to MBR ; write data on MBR to memory address 0x402 ; write 0x00 to MBR ; read address 0x402, store data to MBR ; write MBR to ACC (via BUS) ; write 0x05 to MBR ; compare ACC with MBR, jump to here if greater than ; read ACC, store to MBR ; write data on MBR to memory address 0x403 ; shift left ACC ; read ACC, store to MBR ; write data on MBR to memory at address 0x405
--------------	---	---

instructions

TRACS Assembly Program

ORG 0x010

start	WB 0x05	; write 0x05 to MBR
	WM 0x400	; write data on MBR to memory at address 0x400
	WB 0x03	; write 0x03 to MBR
	WM 0x401	; write data on MBR to memory at address 0x401
	RM 0x400	; read address 0x400, store data to MBR
	WACC	; write MBR to ACC (via BUS)
	RM 0x401	; read data at memory address 0x401, store to MBR
	ADD	; add ACC to MBR (through BUS)
	RACC	; read ACC, store to MBR
	WM 0x402	; write data on MBR to memory address 0x402
	WB 0x00	; write 0x00 to MBR
	RM 0x402	; read address 0x402, store data to MBR
	WACC	; write MBR to ACC (via BUS)
	WB 0x05	; write 0x05 to MBR
	BRGT here	; compare ACC with MBR, jump to here if greater than
	RACC	; read ACC, store to MBR
	WM 0x403	; write data on MBR to memory address 0x403
here	SHL	; shift left ACC
	RACC	; read ACC, store to MBR
	WM 0x405	; write data on MBR to memory at address 0x405
	EOP	

TRACS Assembler Output

- A typical output after an assembly program is assembled is either a binary (.bin), hexadecimal (.hex) or executable (.exe or .com) file.
- For this application, the output of TRACS assembler is neither, but it will output a text file (.txt) with following content format (see Laboratory Exercise #5):

```
ADDR=<address>; BUS=<instruction high byte>; MainMemory<>;  
ADDR=<address+1>; BUS=<instruction low byte>; MainMemory<>;
```




```
ADDR=0x010; BUS=0x03; MainMemory();  
ADDR=0x011; BUS=0x05; MainMemory();  
ADDR=0x012; BUS=0x0C; MainMemory();  
ADDR=0x013; BUS=0x00; MainMemory();  
ADDR=0x014; BUS=0x03; MainMemory();  
ADDR=0x015; BUS=0x03; MainMemory();  
ADDR=0x016; BUS=0x0C; MainMemory();  
ADDR=0x017; BUS=0x01; MainMemory();  
ADDR=0x018; BUS=0x14; MainMemory();  
ADDR=0x019; BUS=0x00; MainMemory();  
ADDR=0x01A; BUS=0x48; MainMemory();  
ADDR=0x01B; BUS=0x00; MainMemory();  
...  
...  
...  
ADDR=0x02C; BUS=0x03; MainMemory();  
ADDR=0x02D; BUS=0x05; MainMemory();  
ADDR=0x02E; BUS=0x90; MainMemory();  
ADDR=0x02F; BUS=0x34; MainMemory();  
ADDR=0x030; BUS=0x58; MainMemory();  
ADDR=0x031; BUS=0x00; MainMemory();  
ADDR=0x032; BUS=0x0C; MainMemory();  
ADDR=0x033; BUS=0x03; MainMemory();  
ADDR=0x034; BUS=0xB0; MainMemory();  
ADDR=0x035; BUS=0x00; MainMemory();  
ADDR=0x036; BUS=0x58; MainMemory();  
ADDR=0x037; BUS=0x00; MainMemory();  
ADDR=0x038; BUS=0x0C; MainMemory();  
ADDR=0x039; BUS=0x05; MainMemory();  
ADDR=0x03A; BUS=0xF8; MainMemory();  
ADDR=0x03B; BUS=0x00; MainMemory();
```

```

ADDR=0x010; BUS=0x03; MainMemory();
ADDR=0x011; BUS=0x05; MainMemory();
ADDR=0x012; BUS=0x0C; MainMemory();
ADDR=0x013; BUS=0x00; MainMemory();
ADDR=0x014; BUS=0x03; MainMemory();
ADDR=0x015; BUS=0x03; MainMemory();
ADDR=0x016; BUS=0x0C; MainMemory();
ADDR=0x017; BUS=0x01; MainMemory();
ADDR=0x018; BUS=0x14; MainMemory();
ADDR=0x019; BUS=0x00; MainMemory();
ADDR=0x01A; BUS=0x48; MainMemory();
ADDR=0x01B; BUS=0x00; MainMemory();
...
...
...
ADDR=0x02C; BUS=0x03; MainMemory();
ADDR=0x02D; BUS=0x05; MainMemory();
ADDR=0x02E; BUS=0x90; MainMemory();
ADDR=0x02F; BUS=0x34; MainMemory();
ADDR=0x030; BUS=0x58; MainMemory();
ADDR=0x031; BUS=0x00; MainMemory();
ADDR=0x032; BUS=0x0C; MainMemory();
ADDR=0x033; BUS=0x03; MainMemory();
ADDR=0x034; BUS=0xB0; MainMemory();
ADDR=0x035; BUS=0x00; MainMemory();
ADDR=0x036; BUS=0x58; MainMemory();
ADDR=0x037; BUS=0x00; MainMemory();
ADDR=0x038; BUS=0x0C; MainMemory();
ADDR=0x039; BUS=0x05; MainMemory();
ADDR=0x03A; BUS=0xF8; MainMemory();
ADDR=0x03B; BUS=0x00; MainMemory();

```

program start address is 0x010 as defined by the ORG directive

```
ADDR=0x010; BUS=0x03; MainMemory();
ADDR=0x011; BUS=0x05; MainMemory();
ADDR=0x012; BUS=0x0C; MainMemory();
ADDR=0x013; BUS=0x00; MainMemory();
ADDR=0x014; BUS=0x03; MainMemory();
ADDR=0x015; BUS=0x03; MainMemory();
ADDR=0x016; BUS=0x0C; MainMemory();
ADDR=0x017; BUS=0x01; MainMemory();
ADDR=0x018; BUS=0x14; MainMemory();
ADDR=0x019; BUS=0x00; MainMemory();
ADDR=0x01A; BUS=0x48; MainMemory();
ADDR=0x01B; BUS=0x00; MainMemory();
...
...
...
ADDR=0x02C; BUS=0x03; MainMemory();
ADDR=0x02D; BUS=0x05; MainMemory();
ADDR=0x02E; BUS=0x90; MainMemory();
ADDR=0x02F; BUS=0x34; MainMemory();
ADDR=0x030; BUS=0x58; MainMemory();
ADDR=0x031; BUS=0x00; MainMemory();
ADDR=0x032; BUS=0x0C; MainMemory();
ADDR=0x033; BUS=0x03; MainMemory();
ADDR=0x034; BUS=0xB0; MainMemory();
ADDR=0x035; BUS=0x00; MainMemory();
ADDR=0x036; BUS=0x58; MainMemory();
ADDR=0x037; BUS=0x00; MainMemory();
ADDR=0x038; BUS=0x0C; MainMemory();
ADDR=0x039; BUS=0x05; MainMemory();
ADDR=0x03A; BUS=0xF8; MainMemory();
ADDR=0x03B; BUS=0x00; MainMemory();
```

program start address is 0x010 as defined by the ORG directive

BRGT instruction, jump to “here” which is address 0x034

```

ADDR=0x010; BUS=0x03; MainMemory();
ADDR=0x011; BUS=0x05; MainMemory();
ADDR=0x012; BUS=0x0C; MainMemory();
ADDR=0x013; BUS=0x00; MainMemory();
ADDR=0x014; BUS=0x03; MainMemory();
ADDR=0x015; BUS=0x03; MainMemory();
ADDR=0x016; BUS=0x0C; MainMemory();
ADDR=0x017; BUS=0x01; MainMemory();
ADDR=0x018; BUS=0x14; MainMemory();
ADDR=0x019; BUS=0x00; MainMemory();
ADDR=0x01A; BUS=0x48; MainMemory();
ADDR=0x01B; BUS=0x00; MainMemory();
...
...
...
ADDR=0x02C; BUS=0x03; MainMemory();
ADDR=0x02D; BUS=0x05; MainMemory();
ADDR=0x02E; BUS=0x90; MainMemory();
ADDR=0x02F; BUS=0x34; MainMemory();
ADDR=0x030; BUS=0x58; MainMemory();
ADDR=0x031; BUS=0x00; MainMemory();
ADDR=0x032; BUS=0x0C; MainMemory();
ADDR=0x033; BUS=0x03; MainMemory();
ADDR=0x034; BUS=0xB0; MainMemory();
ADDR=0x035; BUS=0x00; MainMemory();
ADDR=0x036; BUS=0x58; MainMemory();
ADDR=0x037; BUS=0x00; MainMemory();
ADDR=0x038; BUS=0x0C; MainMemory();
ADDR=0x039; BUS=0x05; MainMemory();
ADDR=0x03A; BUS=0xF8; MainMemory();
ADDR=0x03B; BUS=0x00; MainMemory();

```

program start address is 0x010 as defined by the ORG directive

BRGT instruction, jump to “here” which is address 0x034

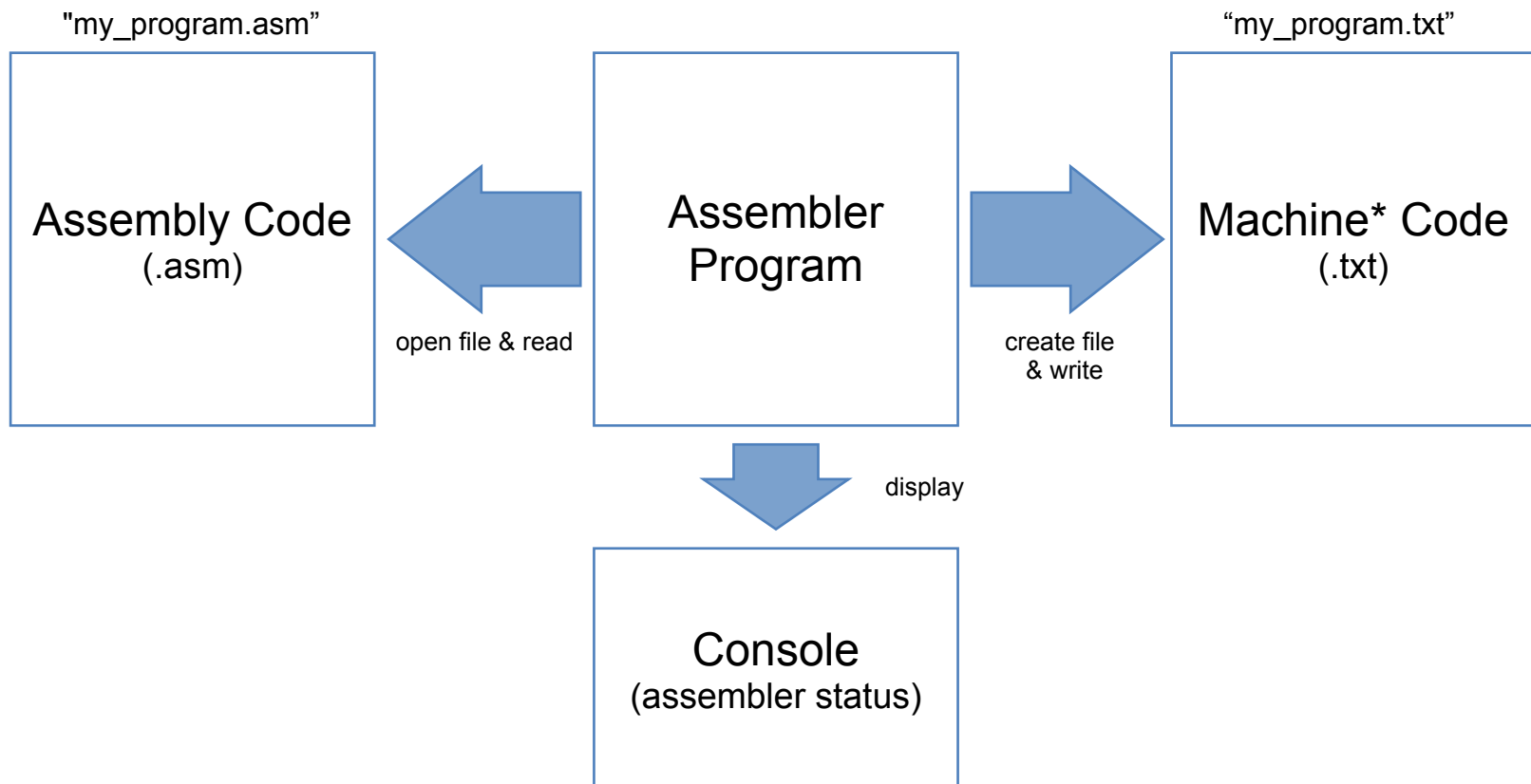
EOP instruction, no more instruction below

Basic Error Detection

- Illegal operand - error occurs when an instruction has an operand when it should not have
- Illegal address - error when an operand (address) is out of range*
- No EOP - error when a program does not have an EOP
- Unknown instruction - error when an instruction is not part of the instruction set
- Errors should be reported on the output console otherwise report as build success.

* data memory start address is implied to be at 0x400 until 0x7FF and I/O memory is at 0x000 to 0x01F

Assembly Process





University of San Carlos | Department of
COMPUTER ENGINEERING

CpE 3202
Computer Organization & Architecture

End of Briefing

Note: This material was written by Van B. Patiluna (USC). All contents contributed are copyright to the respective authors. For instructional use only. **Do not distribute.**

References:

- <https://searchdatacenter.techtarget.com/definition/assembler>
- <https://www.ibm.com/docs/en/zos/2.1.0?topic=introduction-assembler-program>