## Chapter 5

## Assembly Language



## Assembly



## Computer Systems firtu eortion

## Two types of bit patterns

- Instructions
- Mnemonics for opcodes
- Letters for addressing modes
- Data
- Pseudo-ops, also called dot commands


## Computer Systems

| aaa | Addressing Mode | Letters |
| :--- | :--- | :--- |
| 000 | Immediate | i |
| 001 | Direct | d |
| 010 | Indirect | n |
| 011 | Stack-relative | s |
| 100 | Stack-relative deferred | sf |
| 101 | Indexed | x |
| 110 | Stack-indexed | sx |
| 111 | Stack-deferred indexed | sfx |

## Computer Systems

| Instruction Specifier | Mnemonic | Instruction | Addressing Mode | Status <br> Bits |
| :---: | :---: | :---: | :---: | :---: |
| 00000000 | STOP | Stop execution | U |  |
| 00000001 | RET | Return from CALL | U |  |
| 00000010 | RETTR | Return from trap | U |  |
| 00000011 | MOVSPA | Move SP to A | U |  |
| 00000100 | MOVFLGA | Move NZVC flags to A $\langle 12 . .15\rangle$ | U |  |
| 00000101 | MOVAFLG | Move A $\langle 12.15\rangle$ to NZVC flags | $\cup$ |  |
| $0000011 r$ | NOTr | Bitwise invert r | U | NZ |
| 0000 100r | NEGr | Negate r | U | NZV |
| 0000 101r | ASLr | Arithmetic shift left r | U | NZVC |
| 0000 110r | AsRr | Arithmetic shift right r | U | NZC |
| $0000111 r$ | ROLr | Rotate left r | U | C |
| 0001 000r | RORr | Rotate right r | U | C |


| 0001 001a | BR | Branch unconditional | $i, x$ |
| :--- | :--- | :--- | :--- |
| 0001 010a | BRLE | Branch if less than or equal to | $i, x$ |
| 0001 011a | BRLT | Branch if less than | $i, x$ |
| 0001 100a | BREQ | Branch if equal to | $i, x$ |
| 0001 101a | BRNE | Branch if not equal to | $i, x$ |
| 0001 110a | BRGE | Branch if greater than or equal to | $i, x$ |
| 0001 111a | BRGT | Branch if greater than | $i, x$ |
| 0010 000a | BRV | Branch if $V$ | $i, x$ |
| 0010 001a | BRC | Branch if C | $i, x$ |
| 0010 010a | CALL | Call subroutine | $i, x$ |
| 0010 011n | NOPn | Unary no operation trap | U |
| 0010 1aaa | NOP | Nonunary no operation trap | $i$ |


| 0011 Oaaa | DECI | Decimal input trap | $d, n, s, s f, x, s x, s f x$ | NZV |
| :--- | :--- | :--- | :--- | :--- |
| 0011 1aaa | DECO | Decimal output trap | $i, d, n, s, s f, x, s x, s f x$ |  |
| 0100 0aaa | HEXO | Hexadecimal output trap | $i, d, n, s, s f, x, s x, s f x$ |  |
| 0100 1aaa | STRO | String output trap | $d, n, s, s f, x$ |  |
| 0101 0aaa | ADDSP | Add to stack pointer (SP) | $i, d, n, s, s f, x, s x, s f x$ | NZVC |
| 0101 1aaa | SUBSP | Subtract from stack pointer (SP) | $i, d, n, s, s f, x, s x, s f x$ | NZVC |


| 0110 raaa | ADDr | Add to r | $i, d, n, s, s f, x, s x, s f x$ | NZVC |
| :---: | :---: | :---: | :---: | :---: |
| 0111 raaa | SUBr | Subtract from r | $i, d, n, s, s f, x, s x, s f x$ | NZVC |
| 1000 raaa | ANDr | Bitwise AND to r | $i, d, n, s, s f, x, s x, s f x$ | NZ |
| 1001 raaa | ORr | Bitwise OR to r | $i, d, n, s, s f, x, s x, s f x$ | NZ |
| 1010 raaa | CPWr | Compare word to r | $i, d, n, s, s f, x, s x, s f x$ | NZVC |
| 1011 raaa | CPBr | Compare byte to r $\langle 8 . .15\rangle$ | $i, d, n, s, s f, x, s x, s f x$ | NZVC |
| 1100 raaa | LDWr | Load word r from memory | $i, d, n, s, s f, x, s x, s f x$ | NZ |
| 1101 raaa | LDBr | Load byte r $\langle 8 . .15\rangle$ from memory | $i, d, n, s, s f, x, s x, s f x$ | NZ |
| 1110 raaa | STWr | Store word r to memory | $d, n, s, s f, x, s x, s f x$ |  |
| 1111 raaa | STBr | Store byte r $\langle 8 . .15\rangle$ to memory | $d, n, s, s f, x, s x, s f x$ |  |

## Computer Systems fieru eortion

## The unimplemented opcode instructions

- NOPn Unary no-operation trap
- NOP Nonunary no-operation trap
- DECI Decimal input trap
- DECO Decimal output trap
- HEXO Hexadecimal output trap
- STRO String output trap


## Computer Systems rieru eortion

## Pseudo-operations

- . ADDRSS The address of a symbol
$\begin{array}{ll}\text { - . ALIGN } & \text { Padding to align at a me } \\ \text { - . ASCII } & \text { A string of ASCll bytes }\end{array}$
- . BLOCK A block of zero bytes
- . BURN Initiate ROM burn
- . BYTE A byte value
- .END
- . EQUATE
- .WORD A word value

```
Assembler Input
;Stan Warford
;May 1, 2017
;A program to output "Hi"
;
LDBA OxOOOD,d ;Load byte accumulator 'H'
STBA OxFC16,d ;Store byte accumulator output device
LDBA Ox000E,d ;Load byte accumulator 'i'
STBA OxFC16,d ;Store byte accumulator output device
STOP ;Stop
.ASCII "Hi" ;ASCII "Hi" characters
.END
```


## Assembler Output

```
D1 00 OD F1 FC 16 D1 00 OE F1 FC 16 00 48 69 zz
```


## Program Output

Hi

## Computer Systems



## Computer Systems

Input


# Computer Systems 

```
Assembler Input
LDBA OxFC15,d ;Input first character
STBA Ox0013,d ;Store first character
LDBA OxFC15,d ;Input second character
STBA OxFC16,d ;Output second character
LDBA Ox0013,d ;Load first character
STBA OxFC16,d ;Output first character
STOP ;Stop
.BLOCK 1 ;Storage for first character
```

.END

## Assembler Output

D1 FC 15 F1 0013 D1 FC 15 F1 FC 16 D1 0013 F1
FC 160000 zz

## Program Input <br> up

## Program Output pu

## Computer Systems

```
Assembler Input
LDWA 0x000D,d ;A <- first number
ADDA 0x000F,d ;Add the two numbers
ORA Ox0011,d ;Convert sum to character
STBA OxFC16,d ;Output the character
STOP ;Stop
.WORD 5 ;Decimal 5
.WORD 3 ;Decimal 3
.WORD Ox0030 ;Mask for ASCII char
.END
```


## Assembler Output

C1 00 OD 6100 OF 910011 F1 FC 1600000500
030030 zz

## Program Output

8

## Computer Systems



## Computer Systems

Assembler Input
ldwa 0x000D,d ;A <- first number
ADda 0x000F,d ;Add the two numbers ORA 0x0011, d Convert sum to character
StBA OXfc16 , d ;Output the character
.WORD 5 ;Decimal 5
.worD 3 ;Decimal 3
.WORD Ox0030 ;Mask for ASCII char
.end

## Assembler Listing

| Addr | Object code | Mnemon | Operand | Comment |
| :---: | :---: | :---: | :---: | :---: |
| 0000 | C1000D | LDWA | 0x000D, d | ; $\mathrm{A}<-\mathrm{first}$ number |
| 0003 | 61000F | ADDA | 0x000F, d | ;Add the two numbers |
| 0006 | 910011 | ORA | 0x0011, d | ; Convert sum to character |
| 0009 | F1FC16 | STBA | 0xFC16, d | ;Output the character |
| 000C | 00 | STOP |  | ;Stop |
| 000D | 0005 | .WORD | 5 | ;Decimal 5 |
| 000F | 0003 | . WORD | 3 | ;Decimal 3 |
| 0011 | 0030 | .WORD | 0x0030 | ; Mask for ASCII char |
| 0013 |  | . END |  |  |

## Computer Systems fieru sortion

## Direct addressing

- Oprnd = Mem[OprndSpec]
- Asmb5 letter: d
- The operand specifier is the address in memory of the operand.


## Computer Systems firtu eortion

## Immediate addressing

- Oprnd = OprndSpec
- Asmb5 letter: i
- The operand specifier is the operand.


## Computer Systems

```
LDBA 'H',i ;Output 'H'
STBA 0xFC16,d
LDBA 'i',i ;Output 'i'
STBA 0xFC16,d
STOP
.END
```


## Output

Hi

## Computer Systems riftu gotion

## The decimal input instruction

- Instruction specifier: 0011 Oaaa
- Mnemonic: DECI
- Convert a string of ASCII characters from the input device into a 16-bit signed integer and store it into memory

$$
\text { Oprnd } \leftarrow\{\text { decimal input }\}
$$

## Computer Systems fifth edition

## The decimal output instruction

- Instruction specifier: 001I Iaaa
- Mnemonic: DECO
- Convert a 16 -bit signed integer from memory into a string of ASCII characters and send the string to the output device

$$
\{\text { decimal output }\} \longleftarrow \text { Oprnd }
$$

## Computer Systems rifтu eortion

## The unconditional branch instruction

- Instruction specifier: 0001 001a
- Mnemonic: BR
- Skips to a different memory location for the next instruction to be executed.

$$
\mathrm{PC} \leftarrow\{\mathrm{Oprnd}\}
$$

| 0000 | 120005 | BR | $0 \times 0005$ | ;Branch around data |
| :--- | :--- | :--- | :--- | :--- |
| 0003 | 0000 | IBLOCK | 2 | ;Storage for one integer |

Figure 5.II
(continued)

```
Input
-479
```


## Output

```
\(-479+1=-478\)
```


## Computer Systems fifth edtition

## The string output instruction

- Instruction specifier: 0100 Iaaa
- Mnemonic: STRO
- Send a string of null-terminated ASCII characters to the output device

$$
\{\text { string output }\} \leftarrow \text { Oprnd }
$$

| 0000 | 120005 | BR | 0x0005 | ; Branch around data |
| :---: | :---: | :---: | :---: | :---: |
| 0003 | 0000 | . BLOCK | 2 | ;Storage for one integer |
|  |  | ; |  |  |
| 0005 | 310003 | DECI | 0x0003, ${ }^{\text {d }}$ | ; Get the number |
| 0008 | 390003 | DECO | 0x0003, d | ; and output it |
| 000B | 49001B | STRO | 0x001B, ${ }^{\text {d }}$ | ;Output " + 1 = " |
| 000E | C10003 | LDWA | 0x0003, d | ; $A$ <- the number |
| 0011 | 600001 | ADDA | 1,i | ; Add one to it |
| 0014 | E10003 | STWA | 0x0003, ${ }^{\text {d }}$ | ;Store the sum |
| 0017 | 390003 | DECO | 0x0003, ${ }^{\text {d }}$ | ;Output the sum |
| 001A | 00 | STOP |  |  |
| 001B | 202B20 | . ASCII | " + $1=$ |  |
|  | 31203D |  |  |  |
|  | 2000 |  |  |  |
| 0023 |  | . END |  |  |
| Input |  |  |  |  |
| -479 |  |  |  |  |
| Output |  |  |  |  |
| -479 | $1=-4$ |  |  |  |

## Computer Systems fifth edition

## The hexadecimal output instruction

- Instruction specifier: 0100 0aaa
- Mnemonic: HEXO
- Convert a 2-byte word from memory into four hexadecimal digits and send the string to the output device
$\{$ hexadecimal output $\} \leftarrow$ Oprnd


## Computer Systems rifтu eortion

## Interpreting bit patterns

- Dot commands set bit patterns at assembly time
- Executable statements interpret bit patterns at run time


## Computer Systems

| 0000 | 120009 | BR | 0x0009 | ; Branch around data |
| :---: | :---: | :---: | :---: | :---: |
| 0003 | FFFE | .WORD | 0xFFFE | ; First |
| 0005 | 00 | . BYTE | $0 \times 00$ | ;Second |
| 0006 | 55 | . BYTE | 'U' | ; Third |
| 0007 | 0470 | .WORD | 1136 | ; Fourth |
| ; |  |  |  |  |
| 0009 | 390003 | DECO | 0x0003, d | ; Interpret First as dec |
| 000C | D0000A | LDBA | '\n', i |  |
| 000F | F1FC16 | STBA | 0xFC16, d |  |
| 0012 | 390005 | DECO | 0x0005, d | ; Interpret Second and Third as dec |
| 0015 | D0000A | LDBA | '\n', i |  |
| 0018 | F1FC16 | STBA | 0xFC16, d |  |
| 001B | 410005 | HEXO | 0x0005, d | ; Interpret Second and Third as hex |
| 001E | D0000A | LDBA | '\n', i |  |
| 0021 | F1FC16 | STBA | 0xFC16, d |  |
| 0024 | D10006 | LDBA | 0x0006, d | ; Interpret Third as char |
| 0027 | F1FC16 | STBA | 0xFC16, d |  |
| 002A | D10008 | LDBA | 0x0008, d | ; Interpret Fourth as char |
| 002D | F1FC16 | STBA | 0xFC16, d |  |
| 0030 | 00 | STOP |  |  |
| 0031 |  | . END |  |  |

Figure 5.13
(continued)

## Output <br> -2 <br> 85 <br> 0055 <br> Up

## Computer Systems rifтu eortion

## Disassembler

- The inverse mapping of an assembler is not unique
- Given a bit pattern at level ISA3, you cannot determine the Asmb5 statement that produced it


## Computer Systems

| Assembly Language Program |  |  |  |
| :--- | :--- | :--- | :--- |
| 0000 | D10013 | LDBA | Ox0013,d |
| 0003 | F1FC16 | STBA | OxFC16,d |
| 0006 | D10014 | LDBA | Ox0014, d |
| 0009 | F1FC16 | STBA | OxFC16,d |
| $000 C$ | D10015 | LDBA | Ox0015,d |
| $000 F$ | F1FC16 | STBA | OxFC16,d |
| 0012 | 00 | STOP |  |
| 0013 | $50756 E$ | .ASCII | "Pun" |
| 0016 |  | .END |  |

## Assembly Language Program

| 0000 | D10013 | LDBA | Ox0013,d |
| :--- | :--- | :--- | :--- |
| 0003 | F1FC16 | STBA | OxFC16, d |
| 0006 | D10014 | LDBA | Ox0014,d |
| 0009 | F1FC16 | STBA | OxFC16,d |
| $000 C$ | D10015 | LDBA | Ox0015,d |
| $000 F$ | F1FC16 | STBA | OxFC16,d |
| 0012 | 00 | STOP |  |
| 0013 | $50756 E$ | ADDSP | Ox756E, i |
| 0016 |  | .END |  |

## Program Output

 Pun
## Computer Systems firtu eortion

## Mappings

- The mapping from Asmb5 to ISA3 is one-toone
- The mapping from HOL6 to Asmb5 is one-to-many


## Computer Systems firtu eortion

## Symbols

- Defined by an identifier followed by a colon at the start of a statement
- The value of a symbol is the address of the object code generated by the statement


# Computer Systems <br> FIFTH EDITION 

Figure 5.15

Assembler Listing


# Computer Systems 

Figure 5.15 (continued)

```
Symbol table
\begin{tabular}{|c|c|c|c|}
\hline Symbol & Value & Symbol & Value \\
\hline
\end{tabular}
\begin{tabular}{llll} 
main & 0005 & msg & \(001 B\) \\
num & 0003 & &
\end{tabular}
Input
-479
Output
-479 + 1 = -478
```


# Computer Systems 

## Assembler Input <br> this: DECO this,d <br> STOP <br> .END

## Assembler Listing

 0000390000 this: DECO this,d 000300 STOP 0004 .END
## Output

14592

## Computer Systems



## Computer Systems


(a) Translation directly to machine language.

(b) Translation to assembly language.

## Computer Systems fifth edition

## Translating printf ()

- Translate string output with STRO
- Translate character output with

STBA charOut,d

- Translate integer output with DECO

```
High-Order Language
#include <stdio.h>
int main() {
    printf("Hello, world!\n");
    return 0;
}
```

Assembly Language
0000490004 STRO msg,d
000300 STOP
0004 48656C msg: .ASCII "Hello, world! $\backslash \mathrm{n} \backslash \mathrm{x} 00$ "
6C6F2C
2077 6F
726C64
210A00
0013 .END

## Output

Hello, world!

## Computer Systems

Input
Processing
Output

```
#include <stdio.h>
int main() {
    printf("Hello, world!\n");
    return 0;
}
```

```
49 00 04
```

49 00 04
0
0
48 65 6C 6C 6F 2C 20
48 65 6C 6C 6F 2C 20
77 6F 72 6C 64 21 0A 00
77 6F 72 6C 64 21 0A 00
ZZ

```
ZZ
```

(a) A compiler that translates directly into machine language.

```
#include <stdio.h>
int main() {
    printf("Hello, world!\n");
    return 0;
}
```

(b) A compiler that translates into assembly language.

(a) A compiler that translates to machine language.

| C identifier |
| :--- |
| for variable name | $\rightarrow$| Pep/9 assembly |
| :--- |
| language symbol |$\rightarrow$| Memory |
| :--- |
| address |

(b) A hypothetical compiler for illustrative purposes.

## Computer Systems rifтu eortion

## Global variables

- Allocated at a fixed location in memory with . BLOCK
- Accessed with direct addressing (d)


## Computer Systems fifth edition

## Assignment statements

- Load the accumulator from the right hand side of the assignment with LDWA or LDBA
- Compute the value of the right hand side of the assignment if necessary
- Store the value to the variable on the left hand side of the assignment with STWA or STBA


## Computer Systems firtu eortion

## Translating scanf ()

- Translate character input with

LDBA charIn,d

- Translate integer input with DECI


## Computer Systems

```
High-Order Language
#include <stdio.h>
char ch;
int j;
int main() {
    scanf("%c %d", &ch, &j);
    j += 5;
    ch++;
    printf("%c\n%d\n", ch, j);
    return 0;
}
```

Figure 5.22
(continued)

| Assembly Language |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| 0000 | 120006 |  | BR | main |  |
| 0003 | 00 | ch: | . BLOCK | 1 | ; global variable \#1c |
| 0004 | 0000 | j : | . BLOCK | 2 | ;global variable \#2d |
|  |  | ; |  |  |  |
| 0006 | D1FC15 | main: | LDBA | charln, d | ;scanf("\%c \%d", \&Ch, \&j) |
| 0009 | F10003 |  | STBA | ch, d |  |
| 000C | 310004 |  | DECI | j,d |  |
| 000F | C10004 |  | LDWA | j,d | ; ${ }^{\text {+ }} 5$ |
| 0012 | 600005 |  | ADDA | 5,i |  |
| 0015 | E10004 |  | STWA | j, d |  |
| 0018 | D10003 |  | LDBA | ch, d | ; ch++ |
| 001B | 600001 |  | ADDA | 1, i |  |
| 001E | F10003 |  | STBA | ch, d |  |
| 0021 | D10003 |  | LDBA | ch, d | ;printf("\%c\n\%d\n", ch, j) |
| 0024 | F1FC16 |  | STBA | charOut, d |  |
| 0027 | D0000A |  | LDBA | '\n', i |  |
| 002A | F1FC16 |  | STBA | charOut, d |  |
| 002D | 390004 |  | DECO | j,d |  |
| 0030 | D0000A |  | LDBA | '\n',i |  |
| 0033 | F1FC16 |  | STBA | charOut, d |  |
| 0036 | 00 |  | STOP |  |  |
| 0037 |  |  | . END |  |  |

Figure 5.22
(continued)
Input
M 419

## Output

N
424

## Computer Systems

```
#include <stdio.h>
char ch;
int j;
int main() {
    scanf("%c %d", &ch, &j);
    j += 5;
    ch++;
    printf("%c\n%d\n", ch, j);
    return 0;
}
\begin{tabular}{c|c|c|c|}
\hline & symbol & value & kind \\
\hline\([0]\) & ch & 0003 & sChar \\
\hline\([1]\) & \(j\) & 0004 & sInt \\
\hline\([2]\) & \(\vdots\) & \(\vdots\) & \(\vdots\) \\
\hline
\end{tabular}
```


## Computer Systems

```
#include <stdio.h>
int j;
float y;
int main () {
    j = j % 8;
    y = y % 8; // Compile error
}
\begin{tabular}{c|c|c|c|}
\hline & symbol & value & kind \\
\hline\([0]\) & \(j\) & 0003 & sInt \\
\hline\([1]\) & \(Y\) & 0005 & sFloat \\
\hline\([2]\) & \(\vdots\) & \(\vdots\) & \(\vdots\) \\
\hline
\end{tabular}
```


## Computer Systems firtu eortion

## Trace tags

- Format trace tags
- Required for global and local variables
- Symbol trace tags
- Not required for global variables


## Computer Systems firtu eortion

## Format trace tags

- \#1c One-byte character
- \#1d One-byte decimal
- \#2d Two-byte decimal
- \#1h One-byte hexadecimal
- \#2h Two-byte hexadecimal


## Computer Systems fifth edition

## The arithmetic shift right instruction

- Instruction specifier: 0000 IIOr
- Mnemonic: ASRr (ASRA, ASRX)
- Performs a one-bit arithmetic shift right on a 16-bit register

$$
\begin{gathered}
\mathrm{C} \leftarrow \mathrm{r}\langle 15\rangle, \mathrm{r}\langle 1 . .15\rangle \leftarrow \mathrm{r}\langle 0 . .14\rangle \\
\mathrm{N} \leftarrow \mathrm{r}<0, \mathrm{Z} \leftarrow \mathrm{r}=0
\end{gathered}
$$

## Computer Systems



## CPU


(a) Before.

CPU

(b) After.

## Computer Systems fifth edition

## The arithmetic shift left instruction

- Instruction specifier: 0000 IO Ir
- Mnemonic: ASLr (ASLA, ASLX)
- Performs a one-bit arithmetic shift left on a 16-bit register

$$
\begin{gathered}
\mathrm{C} \leftarrow \mathrm{r}\langle 0\rangle, \mathrm{r}\langle 0 . .14\rangle \leftarrow \mathrm{r}\langle 1 . .15\rangle, \mathrm{r}\langle 15\rangle \leftarrow 0 \\
\mathrm{~N} \leftarrow \mathrm{r}<0, \mathrm{Z} \leftarrow \mathrm{r}=0, \mathrm{~V} \leftarrow\{\text { overflow }\}
\end{gathered}
$$

## Computer Systems piftu eotrion

## The rotate left instruction

- Instruction specifier: 0000 IIIr
- Mnemonic: ROLr (ROLA, ROLX)
- Performs a one-bit rotate left on a I6-bit register
$\mathrm{C} \leftarrow \mathrm{r}\langle 0\rangle, \mathrm{r}\langle 0 . .14\rangle \leftarrow \mathrm{r}\langle 1 . .15\rangle, \mathrm{r}\langle 15\rangle \leftarrow \mathrm{C} ;$


## Computer Systems fifth edition

## The rotate right instruction

- Instruction specifier: 0001 000r
- Mnemonic: RORr (RORA, RORX)
- Performs a one-bit rotate right on a I6-bit register
$\mathrm{C} \leftarrow \mathrm{r}\langle 15\rangle, \mathrm{r}\langle 1 . .15\rangle \leftarrow \mathrm{r}\langle 0 . .14\rangle, \mathrm{r}\langle 0\rangle \leftarrow \mathrm{C} ;$


## Computer Systems rifтu eortion

## Constants

- Equate the constant to its value with . EQUATE
- . EQUATE does not generate object code
- The value of the constant symbol is not an address


## Computer Systems

```
High-Order Language
#include <stdio.h>
const int bonus = 10;
int exam1;
int exam2;
int score;
int main() {
    scanf("%d %d", &exam1, &exam2);
    score = (exam1 + exam2) / 2 + bonus;
    printf("score = %d\n", score);
    return 0;
}
```

Figure 5.27 (continued)

| Assembly Language |  |  |  |  |  |
| :---: | :---: | :---: | :---: | :---: | :---: |
| 0000 | 120009 |  | BR | main |  |
|  |  | bonus: | . EQUATE | 10 | ; constant |
| 0003 | 0000 | exam1: | . BLOCK | 2 | ; global variable \#2d |
| 0005 | 0000 | exam2: | . BLOCK | 2 | ;global variable \#2d |
| 0007 | 0000 | score: | . BLOCK | 2 | ;global variable \#2d |
|  |  | ; |  |  |  |
| 0009 | 310003 | main: | DECI | exam1, ${ }^{\text {d }}$ | ;scanf("\%d \%d", \&exam1, \&exam2) |
| 000C | 310005 |  | DECI | exam2,d |  |
| 000F | C10003 |  | LDWA | exam1, ${ }^{\text {d }}$ | ;score = (exam1 + exam2) / 2 + bonus |
| 0012 | 610005 |  | ADDA | exam2,d |  |
| 0015 | OC |  | ASRA |  |  |
| 0016 | 60000A |  | ADDA | bonus,i |  |
| 0019 | E10007 |  | STWA | score, d |  |
| 001C | 490029 |  | STRO | msg, d | ;printf("score = \%d\n", score) |
| 001F | 390007 |  | DECO | score, d |  |
| 0022 | D0000A |  | LDBA | '\n',i |  |
| 0025 | F1FC16 |  | STBA | charOut |  |
| 0028 | 00 |  | STOP |  |  |
| 0029 | 73636F | msg : | . ASCII | "score |  |
|  | 726520 |  |  |  |  |
|  | 3D2000 |  |  |  |  |
| 0032 |  |  | . END |  |  |

# Computer Systems 

Figure 5.27
(continued)

| Symbol | Value | Symbol | Value |
| :---: | :---: | :---: | :---: |
| bonus | 000A | exam1 | 0003 |
| exam2 | 0005 | main | 0009 |
| msg | 0029 | score | 0007 |

```
Input
6884
```


## Output <br> score $=86$

## Computer Systems



# Computer Systems 

Figure 5.28
(continued)

## Assembly Language

Symbol table

| Symbol | Value | Symbol | Value |
| :---: | :---: | :---: | :---: |
| bonus | 000A | exam1 | 0020 |
| exam2 | 0022 | main | 0000 |
| msg | 0026 | score | 0024 |

## Computer Systems fieru bortion

Figure 5.29


