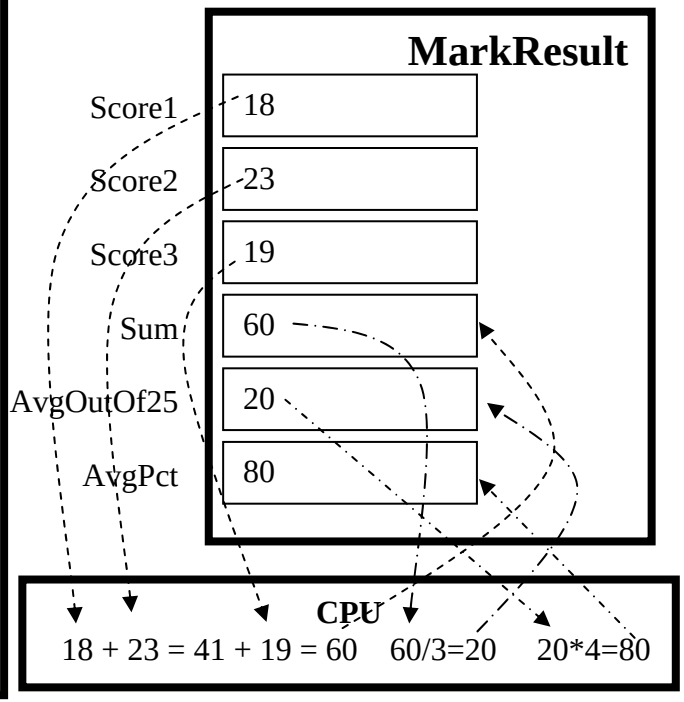
# ITI1120 - Section 4 Exercise Solutions

**Program Memory** **Exercise 4-1 Tracing Example** **Working memory**

Call: AvgPct ← MarkResult(18, 23, 19)

Givens: Score1, Score2, Score3 (scores out of 25)
Results: AvgPct (average of scores, out of 100)
Intermediates:
- Sum (sum of scores)
- AvgOutOf25 (average of scores, out of 25)

Header: AvgPct ← MarkResult( Score1, Score2, Score3 )
Body:
1. Sum ← Score1 + Score2 + Score3
2. AvgOutOf25 ← Sum / 3
3. AvgPct ← AvgOutOf25 * 4




Trace Table for AvgPct ← MarkResult(18, 23, 19)

| Statement | Score1 | Score2 | Score3 | Sum | AvgOutOf25 | AvgPct |
|---|---|---|---|---|---|---|
| Initial values | **18** | **23** | **19** | **?** | **?** | **?** |
| 1. Sum ← Score1 + Score2 + Score3 | | | | **60** | | |
| 2. AvgOutOf25 ← Sum / 3 | | | | | **20** | |
| 3. AvgPct ← AvgOutOf25 * 4 | | | | | | **80** |

**Program Memory** **Exercise 4-2 Tracing a Call** **Working memory**

Givens: none
Results: none
Intermediates:
**First, Second, Third** (three scores)
**Average** (average of scores, out of 100)
Header: M**ain()**
Body:
(Read in scores from the user)

1. **PrintLine("Please enter three scores")**
2. **First ← ReadReal()**
3. **Second ← ReadReal ()**
4. **Third ← ReadReal ()**

(Call the MarkUser algorithm)

5. **Average ← MarkResult(First, Second, Third)**

(Print the average for the user)

6. **PrintLine("The average is ", Average)**

Givens: **Score1, Score2, Score3** (scores out of 25)
Results: **AvgPct** (average of scores, out of 100)
Intermediates: **Sum** (sum of scores)
**AvgOutOf25** (average of scores, out of 25)
Header: **AvgPct ← MarkResult( Score1, Score2, Score3 )**
Body:

1. **Sum ← Score1 + Score2 + Score3**
2. **AvgOutOf25 ← Sum / 3**
3. **AvgPct ← AvgOutOf25 * 4**

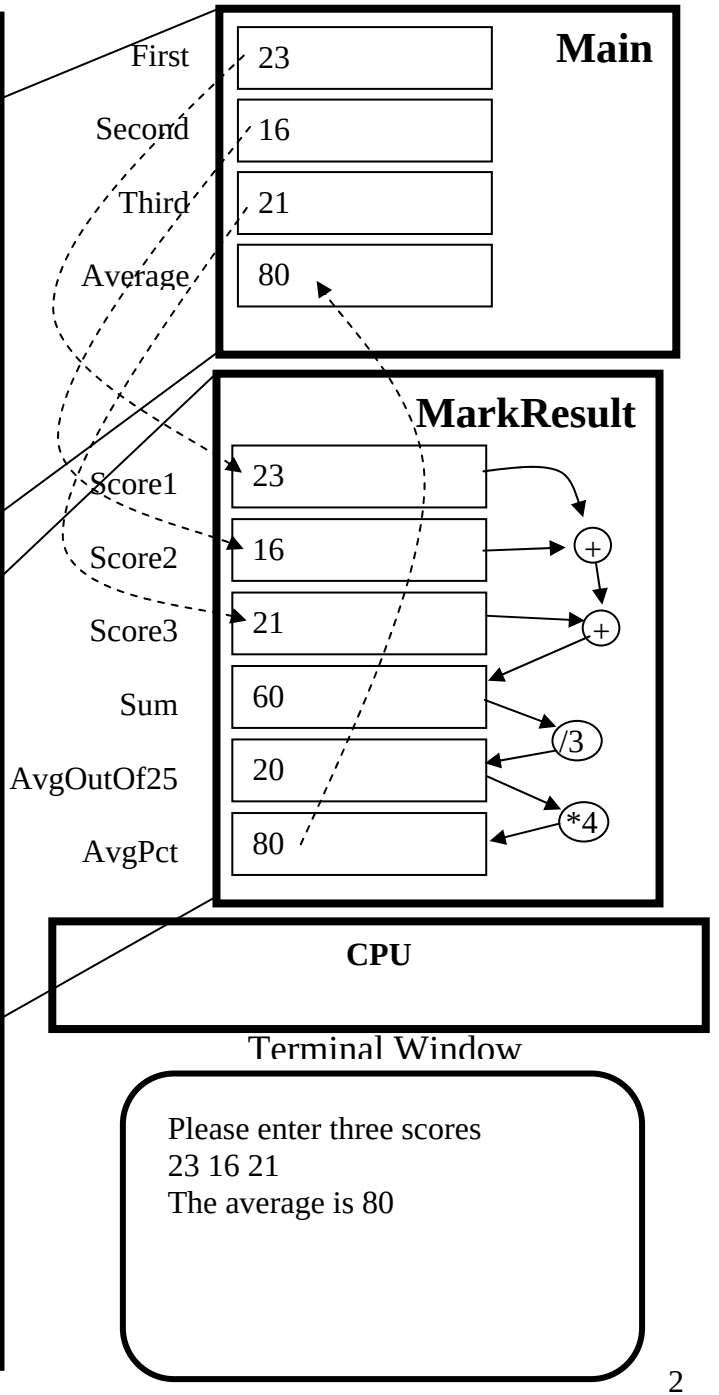


**Table 1** - Table for main algorithm:

Interaction with user:

Please enter three scores out of 25

***23 16 21***

The average is 80 percent

| Statements | First | Second | Third | Average |
|---|---|---|---|---|
| Initial values | **?** | **?** | **?** | **?** |
| 1. PrintLine("Please enter three scores") | | | | |
| 2. First ← ReadReal() | **23** | | | |
| 3. Second ← ReadReal () | | **16** | | |
| 4. Third ← ReadReal () | | | **21** | |
| 5. Call Average ← MarkResult(First, Second, Third) (See Table 2) | | | | |
| 6. PrintLine("The average is ", Average) | | | | |

Call algorithm MarkResult:

Average ← MarkResult (First, Second, Third)

80 23 16 21

AvgPct ← MarkResult (Score1, Score2, Score3)

**Table 2** – Trace for AvgPct ← MarkResult (23,16,21)

| Statement | Score1 | Score2 | Score3 | Sum | AvgOutOf25 | AvgPct |
|---|---|---|---|---|---|---|
| Initial values | **23** | **16** | **21** | **?** | **?** | **?** |
| 1. Sum ← Score1 + Score2 + Score3 | | | | **60** | | |
| 2. AvgOutOf25 ← Sum / 3 | | | | | **20** | |
| 3. AvgPct ← AvgOutOf25 * 4 | | | | | | **80** |

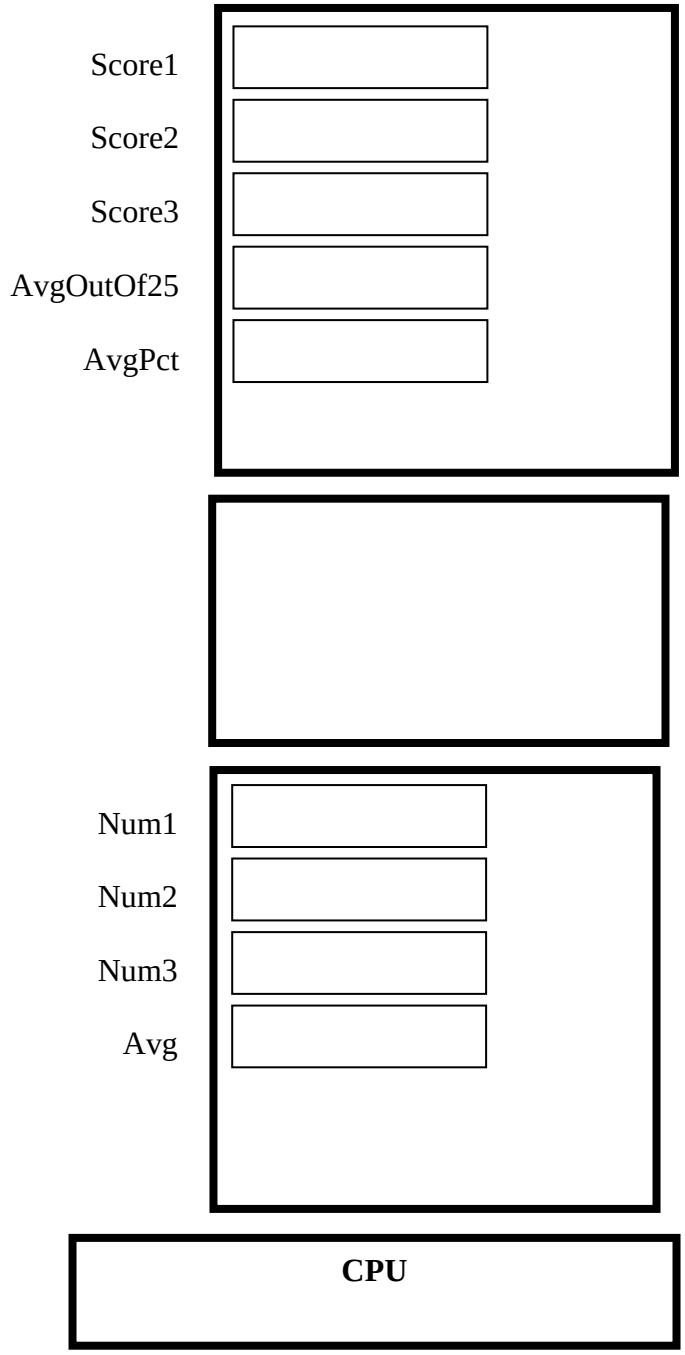
**Program Memory** **Exercise 4-3 – Marks out of 100, again** **Working memory**

Givens: **Score1, Score2, Score3 (scores out of 25)**
Results: **AvgPct (average of scores, out of 100)**
Intermediates:
**AvgOutOf25 (average of scores, out of 25)**
Header: **AvgPct ← MarkResult( Score1, Score2, Score3 )**
Body:

1. **AvgOutOf25 ← Average(Score1, Score2, Score3)**
2. **AvgPct ← AvgOutOf25 * 4**

GIVENS: Num1, Num2, Num3 (three numbers)
RESULTS: Avg (the average of Num1, Num2, and Num3)
HEADER: Avg ← Average(Num1, Num2, Num3)
BODY:

1. Avg ← (Num1 + Num2 + Num3)/3



**Program Memory**

## Exercise 4-4 – Tracing Example

**Working memory**

Call to Average ← MarkResult( 23, 16, 21 )

Givens: Score1, Score2, Score3 (scores out of 25)
Results: AvgPct (average of scores, out of 100)
Intermediates:
AvgOutOf25 (average of scores, out of 25)
Header: AvgPct ← MarkResult( Score1, Score2, Score3 )
Body:
1. AvgOutOf25 ← Average(Score1, Score2, Score3)
2. AvgPct ← AvgOutOf25 * 4

GIVENS: Num1, Num2, Num3 (three numbers)
RESULTS: Avg (the average of Num1, Num2, and Num3)
HEADER: Avg ← Average(Num1, Num2, Num3)
BODY:
1. Avg ← (Num1 + Num2 + Num3)/3

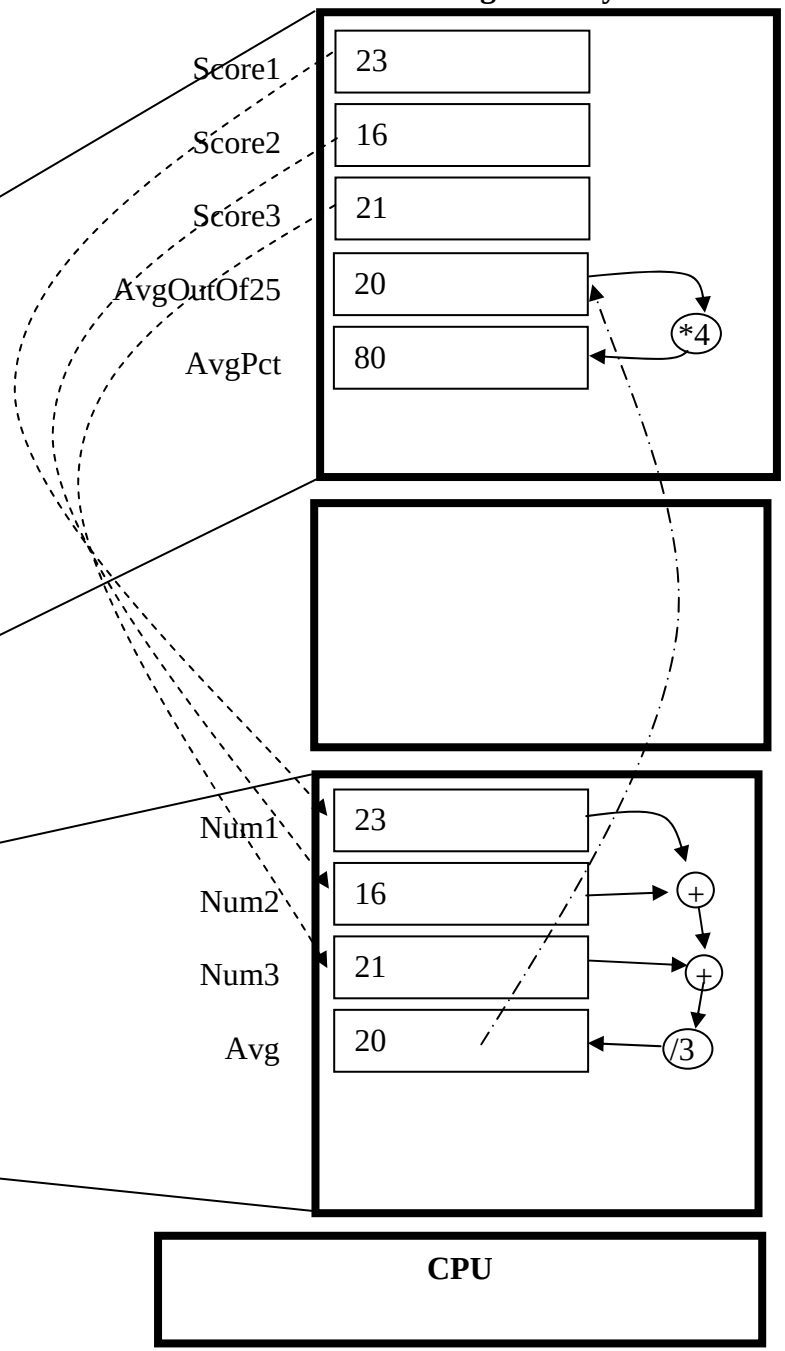


**Table 1** - Table for AvgPct ← MarkResult(23, 16, 21)

| Statements | Score1 | Score2 | Score3 | AvgOutOf25 | AvgPct |
|---|---|---|---|---|---|
| Initial values | **23** | **16** | **21** | **?** | **?** |
| 1. Call AvgOutOf25 ← Average(23, 16, 21) (See Table 2) | | | | **20** | |
| 2. AvgPct ← AvgOutOf25 * 4 | | | | | **80** |

Call algorithm Average:

AvgOutOf25 ← Average (Score1, Score2, Score3)

20 23 16 21

Avg ← Average (Num1, Num2, Num3)

**Table 2** - Table for Avg ← Average(23, 16, 21)

| Statement | Num1 | Num2 | Num3 | Avg |
|---|---|---|---|---|
| Initial values | **23** | **16** | **21** | **?** |
| 1. Sum ← (Num1 + Num2 + Num3)/3 | | | | **20** |

**Program Memory**

## Exercise 4-5 – Reverse Digits

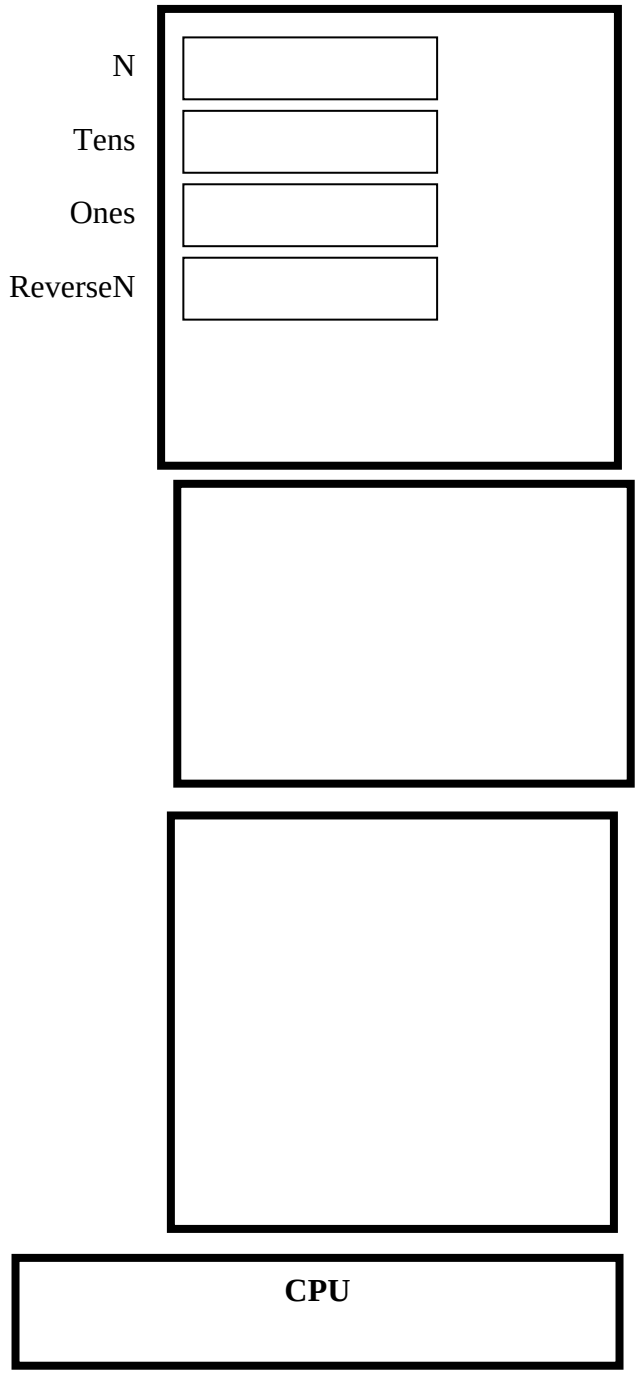
GIVENS: **N (a two digit number)**
RESULTS: **ReverseN (Same digits as N with reverse order)**
IINTERMEDIATES:
**Tens, Ones (N's left and right digit)**
HEADER: **ReverseN ← Rev2(N)**
BODY:

1. **(Tens, Ones) ← Digits(N)**
2. **ReverseN ← 10*Ones + Tens**

The following algorithm is available to extract the ten's and one's digits from a two digit number:

(High, Low) ← Digits( X )

**Working memory**



**Program Memory** **Exercise 4-6 Trace Reverse Digits** **Working memory**

Trace for N = 42, i.e. ReverseN ← Rev2(42)

GIVENS: N (a two digit number)
RESULTS: ReverseN (Same digits as N with reverse order)
IINTERMEDIATES:
Tens, Ones (N's left and right digit)
HEADER: ReverseN ← Rev2(N)
BODY:
3. (Tens, Ones) ← Digits(N)
4. ReverseN ← 10*Ones + Tens

The following algorithm is available to extract the ten's and one's digits from a two digit number:
(High, Low) ← Digits( X )

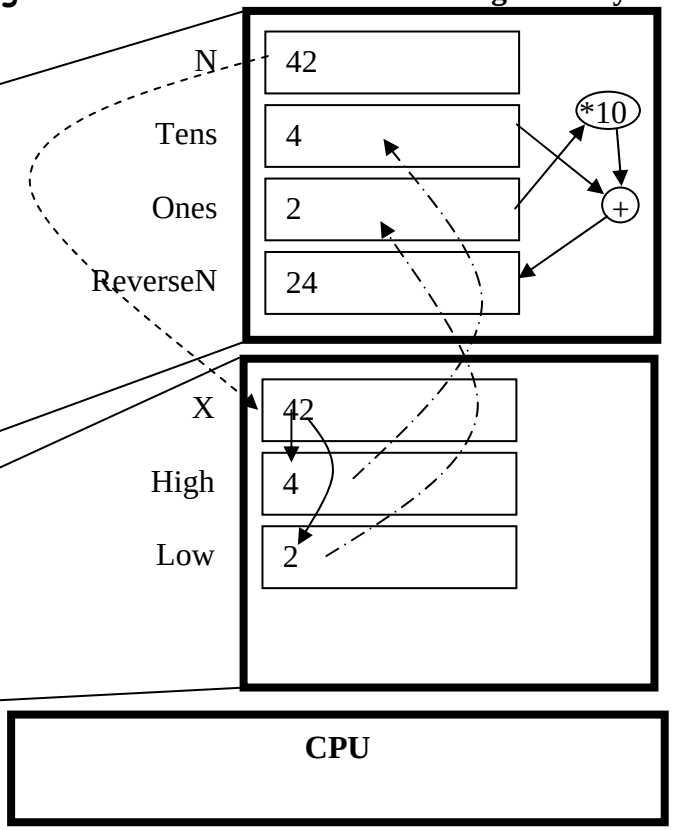



Table 1 - Trace for ReverseN ← Rev2(42))

| Statements | N | Tens | Ones | ReverseN |
|---|---|---|---|---|
| Initial values | **42** | **?** | **?** | **?** |
| 1. Call Digits(N) | | **4** | **2** | |
| 2. ReverseN ← Ones * 10 + Tens | | | | **24** |

Call to (Tens, Ones) ← Digits (N)
(Tens, Ones) ← Digits (N)
4 2 42
(High, Low) ← Digits (X)

**Program Memory** **Exercise 4-7 – Join Four Numbers** **Working memory**

Givens: **W, X, Y, Z** **(positive integers)**
Result: **AllJoined** **(the result of joining W,X,Y, and Z)**
Intermediates:
**WX** **(the results of joining W and X)**
**YZ** **(the result of joining Y and Z)**
Header: **AllJoined ←Join4(W,X,Y,Z)**
Body:

**WX ← Join(W,X)**
**YZ ← Join(Y,Z)**
**AllJoined ← Join (WX, YZ)**

You may assume there is available an algorithm:

C ← Join( A, B )

Givens: A, B, two positive integers
Result: C is the number having the digits in A followed by the digits in B.