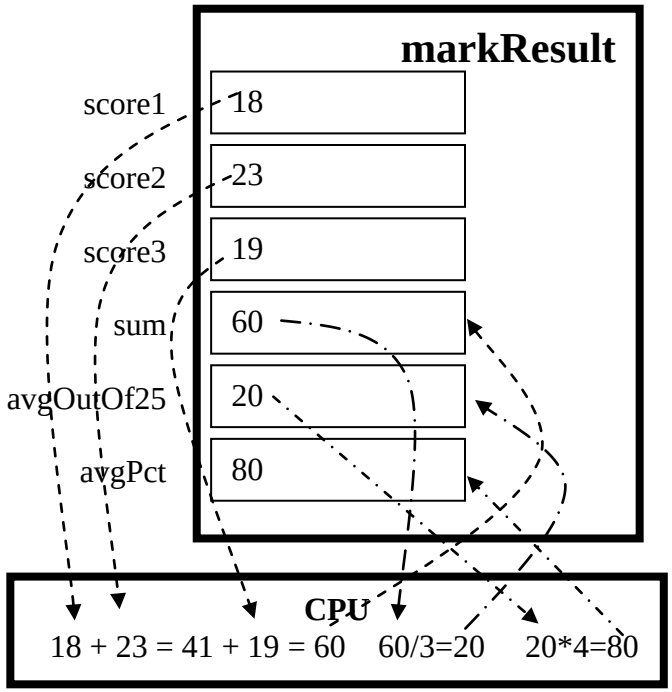
ITI1120 - Section 4 Exercise Solutions

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: **main()**
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**

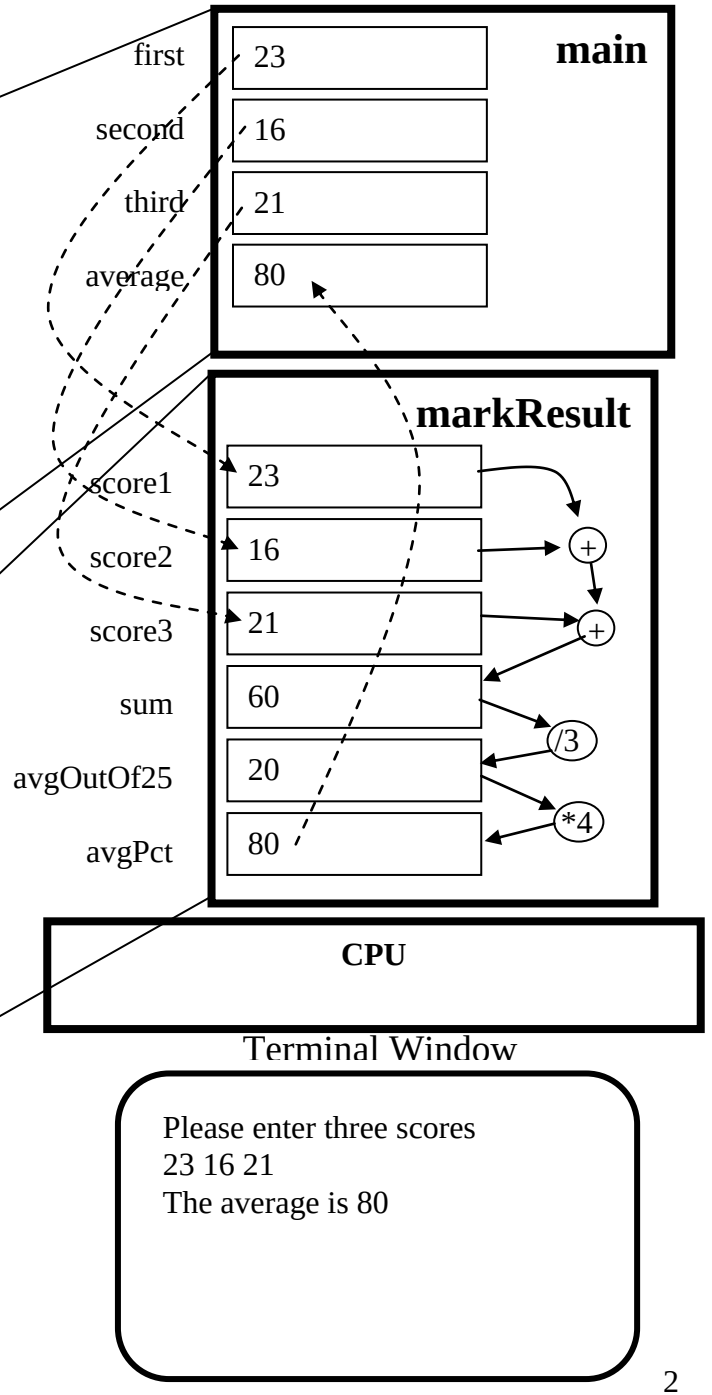


Table 1 - Table for main algorithm:

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**

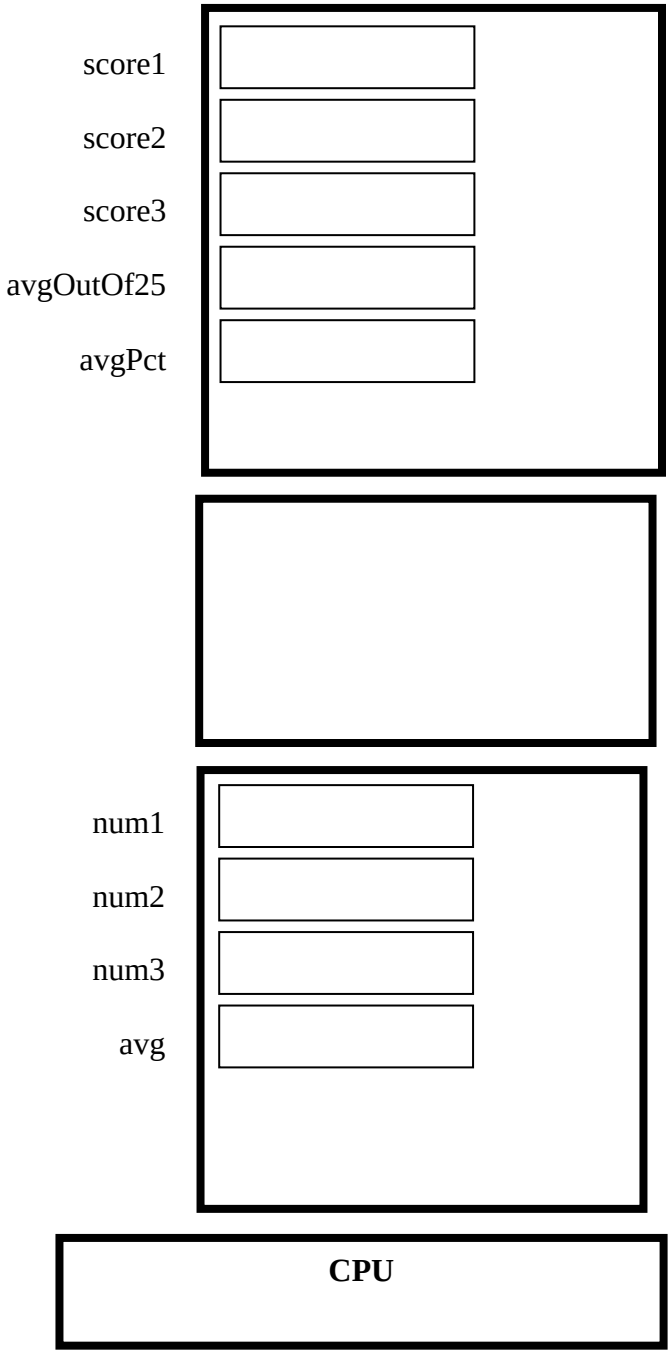
Program Memory **Exercise 4-3 – Marks out of 100, again** **Working memory**

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

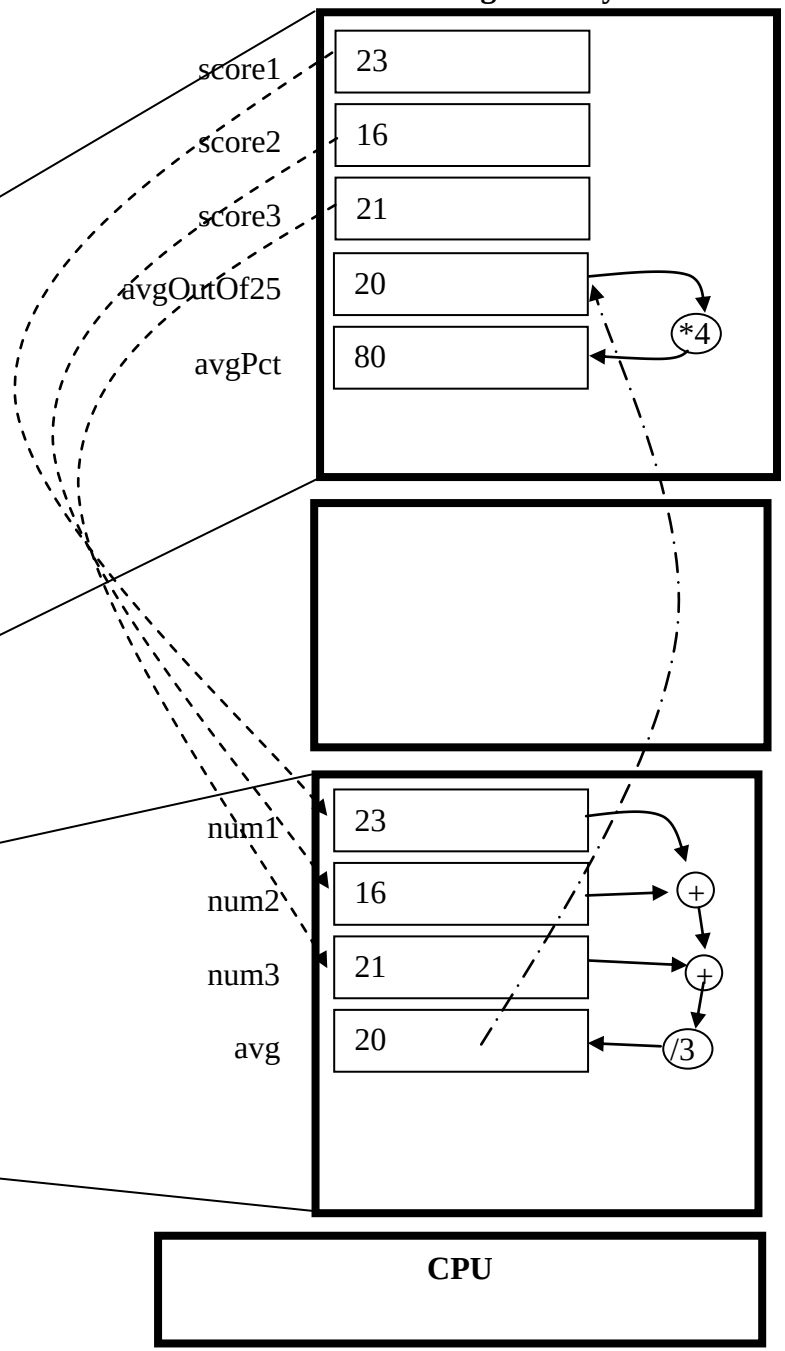


Table 1 - Table for avgPct ← markResult(23, 16, 21)

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

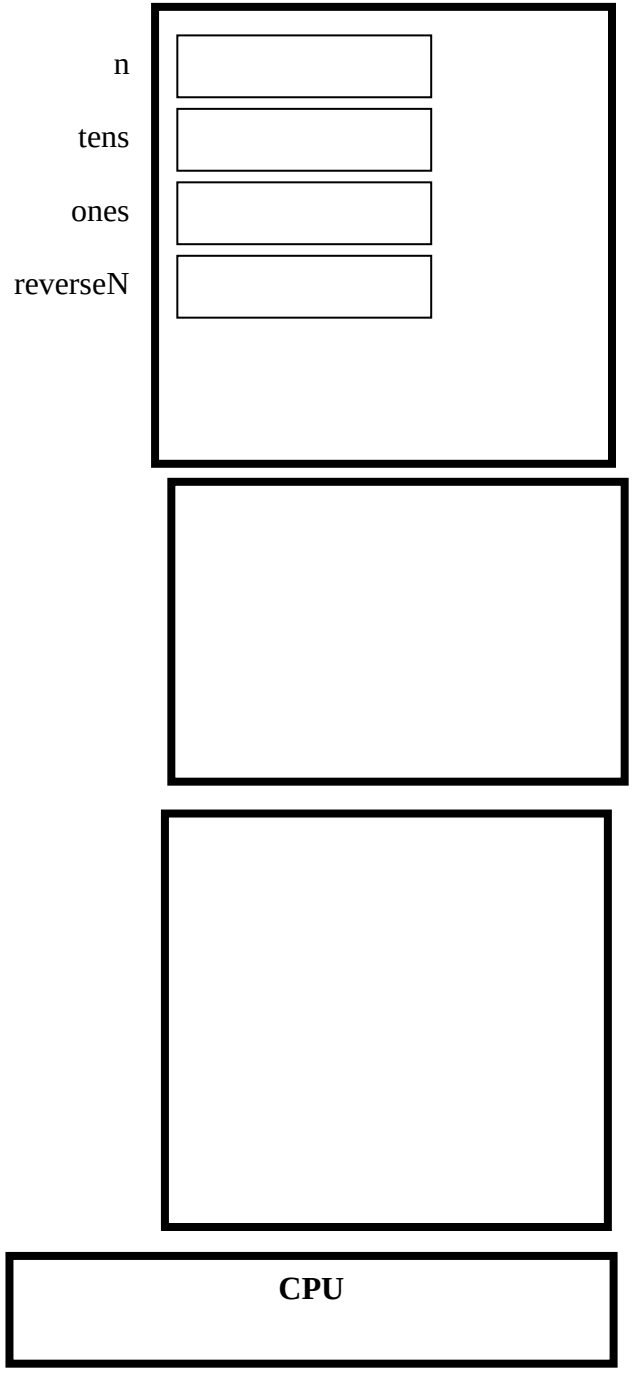
GIVENS: **n (a two digit number)**
RESULTS: **reverseN (Same digits as n with reverse order)**
IINTERMEDIATES:
tens, ones (n's left and right digit)
HEADER: **reverseN ← rev2(n)**
BODY:

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)

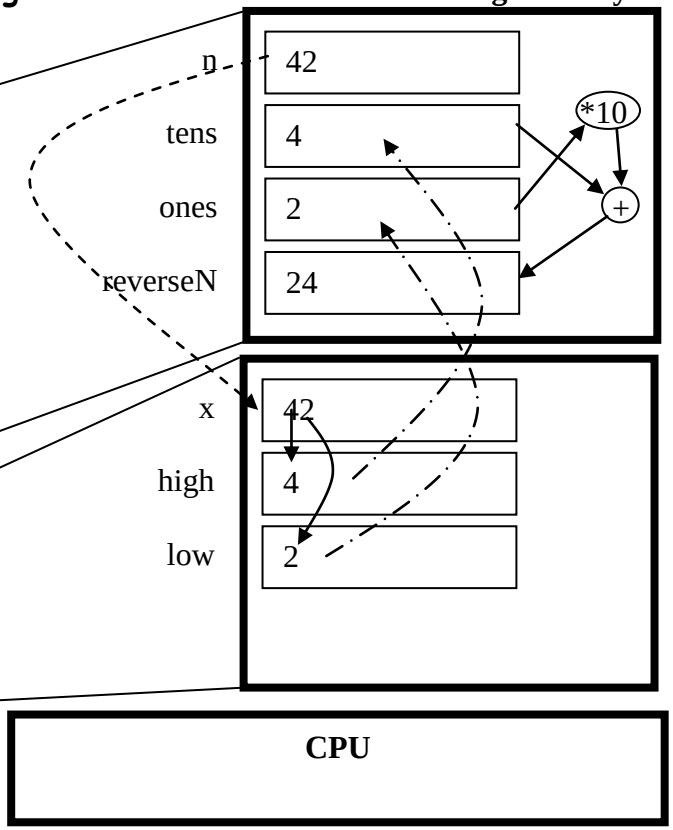


Table 1 - Trace for reverseN ← rev2(42))

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**

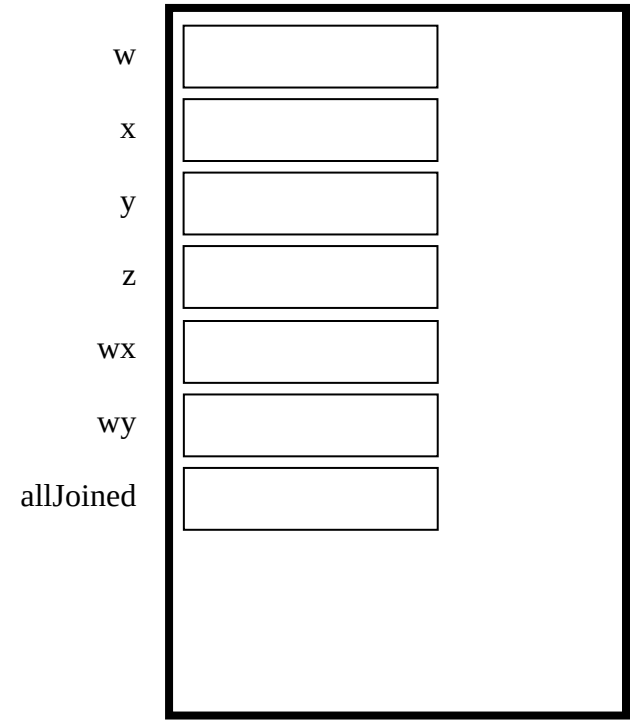
Givens: **w, x, y, z (positive integers)**
Result: **allJoined (the result of joining w,x,y, and z)**
Intermediates:
wx (the results of joining w and x)
yz (the result of joining y and z)
Header: **allJoined ←join4(w,x,y,z)**
Body:

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.



CPU