a) El único sitio lógico para la inserción de la instrucción `NOP` es a continuación de la instrucción `BEQZ` debido a que ésta es la que causa el riesgo de control. Ya que el retardo de salto que se produce en la segmentación de ASG es de un ciclo, la secuencia de instrucciones tendría el siguiente aspecto:

```
i1:  LD     R1, X(R7)
i2:  ADDI   R1, R1, #1
i3:  BEQZ   Rtest, i7
i4:   NOP
i5:  ADD    R2, R1, R2
i6:  SUB    R2, R2, R3
i7:  SD     0(R8),R5
```

Si el salto no fuese efectivo y Rtest es distinto de R1 tendríamos la siguiente Segmentación (**Det**: Detención):

i1: LD R1, X(R7)	IF	ID	EX	MEM	WB					
i2: ADDI R1, R1, #1		IF	ID	Det	EX	MEM	WB			
i3: BEQZ Rtest, i7			IF	Det	ID	EX	MEM	WB		
i4: NOP					**IF**	**ID**	**EX**	**MEM**	**WB**	
i5: ADD R2, R1, R2						IF	ID	EX	MEM	WB
i6: SUB R2, R2, R3							IF	ID	EX	MEM WB
i7: SD 0(R8),R5								IF	ID	EX MEM WB

y si fuese efectivo:

i1: LD R1, X(R7)	IF	ID	EX	MEM	WB				
i2: ADDI R1, R1, #1		IF	ID	Det	EX	MEM	WB		
i3: BEQZ Rtest, i7			IF	Det	ID	EX	MEM	WB	
i4: NOP					**IF**	**ID**	**EX**	**MEM**	**WB**
i7: SD 0(R8),R5						IF	ID	EX	MEM WB

b)

El que `Rtest` sea igual a `R4` no depende de ninguna instrucción previa, por lo tanto, la instrucción `BEQZ` puede ser situada antes de la primera instrucción de suma. La secuencia quedaría como:

```
i1:  LD     R1, X(R7)
i2:  BEQZ   R4, i6
i3:  ADDI   R1, R1, #1
i4:  ADD    R2, R1, R2
i5:  SUB    R2, R2, R3
i6:  SD     0(R8),R5
```

Si el salto no fuese efectivo tendríamos la siguiente segmentación:

i1: LD R1, X(R7)	IF	ID	EX	MEM	WB				
i2: BEQZ R4, i6		IF	ID	EX	MEM	WB			
i3: ADDI R1, R1, #1			**IF**	**ID**	**EX**	**MEM**	**WB**		
i4: ADD R2, R1, R2				IF	ID	EX	MEM	WB	
i5: SUB R2, R2, R3					IF	ID	EX	MEM	WB
i6: SD 0(R8),R5						IF	ID	EX	MEM WB

y si fuese efectivo

i1: LD R1, X(R7)	IF	ID	EX	MEM	WB			
i2: BEQZ R4, i6		IF	ID	EX	MEM	WB		
i3: ADDI R1, R1, #1			**IF**	**ID**	**EX**	**MEM**	**WB**	
i6: SD 0(R8),R5				IF	ID	EX	MEM	WB

c)

Que `Rtest` t sea igual a `R1` depende de la instrucción de suma situada inmediatamente antes de BEQZ, por lo que no hay optimización posible y el código resultante es el mismo que en el apartado (a):

```
i1:  LD     R1, X(R7)
i2:  ADDI   R1, R1, #1
i3:  BEQZ   R1, i7
i4:   NOP
i5:  ADD    R2, R1, R2
i6:  SUB    R2, R2, R3
i7:  SD     0(R8),R5
```