Aluno(s)	Estrutura geral, programa principal e interação com usuário (4,0 pontos)	Subrotina recursiva (3,0 pontos)	Documentação: telas, comentários, explicações (3,0 pontos)	Nota
G1 - Alison	O programa principal está mal organizado com parte dele antes da rotina binary_search e parte dele misturado fisicamente com o código da rotina. Ao executar, o programa interage com o usuário corretamente e gera o resultado correto, terminando após a interação. Para realizar mais de um teste, o programa deve ser rodado a cada vez. A documentação no geral é pobre, não definindo, por exemplo, a alocação de informações que se coloca/retira na/da pilha. (3,0)	A subrotina não é recursiva, violando o principal requisito da especificação do trabalho. Existe um laço disfarçado que implementa o cálculo usando jal operando, sendo que o rótulo operando está na linha imediatamente abaixo, o que não faz qualquer sentido. A subrotina recebe alguns dados via pilha quando chamada pelo programa principal, mas acessa o array diretamente e não por um informação comunicada via pilha, e retorna o resultado via um registrador, ao invés de retornar via pilha, mais uma vez violando (duas vezes) a especificação. (1,0)	Fora uma exposição breve da implementação não-recursiva, apenas respondeu às questões do enunciado, algumas de forma incorreta. Por exemplo, como a subrotina não é recursiva não há uma terceira chamada aninhada da recursão, fazendo com que as respostas das questões 3, 4 e 5 sejam falsas. As telas detalhando a operação da pilha não foram mostradas. (1,5)	5,50
G2 - André, GabrielR e Vermon	O programa principal foi desenvolvido de forma rígida demais, deixando a chave como uma variável interna deste. Para executar o sofware sobre uma chave diferente daquela do código fonte entregue, deve-se editar o código fonte em assembly do MIPS, mudar o valor da variável "chave", remontar o código executável e re-executar o programa, o que é bem ruim. Ou seja, não há interação com o usuário que executa o programa. Isto torna o programa principal um tanto simples e óbvio, sem que os alunos tenham exercitado o uso mais extenso de chamadas do sistema. (3,3)	A subrotina foi implementada de forma precisa, refletindo exatamente o algoritmo informado na especificação. Ótimo!. (3,0)	A documentação descreve a solução usada de forma detalhada e precisa. As ilustrações estão corretas e muito claras. Excelente!! (3,0)	9,30
G3 - Bernado, Luana e Natália	O programa principal foi desenvolvido de forma rígida demais, deixando os valores Prim, Ult e Chave como constantes carregadas pelo programa principal em registradores no início de sua execução. Para executar o sofware sobre uma chave diferente daquela do código fonte entregue, deve-se editar o código fonte em assembly do MIPS, os valores de "Chave", "Prim" e "Ult", remontar o código executável e re-executar o programa, o que é bem ruim. Ou seja, não há interação com o usuário que executa o programa. Isto torna o programa principal um tanto simples e óbvio, sem que os alunos tenham exercitado o uso mais extenso de chamadas do sistema. Além disto a resposta gerada vem desacompanhada de qualquer texto explicativo, tornando a execução do programa críptica. (2,8)	A subrotina até é recursiva, mas possui várias falhas na sua estrutura. Primeiro, os trechos em linguagem de montagem correspondendo aos rótulos "valido", "percorre_array" e "testa_chave" violam o algoritmo recursivo criando laço(s) que percorre o vetor "A" sem nenhuma necessidade. Além disto existe apenas uma chamada recursiva dentro da binary_search e não duas como deveria existir se seguissem o algoritmo original. (2,0)	Documentação um tanto pobre, apenas responderam às questões do enunciado e mostraram o código do programa principal, sem justificar porque o código de binary_search é tão diferente do algoritmo fornecido como base em C. (2,0)	6,80
G4 - Brenda e JenniferAKB	O programa usa vetores pré-determinados para os parâmetros prim/ult/chave, o que é mais limitado que interagir diretamente com o usuário, mas facilita a escrita do programa. Há 5 elementos nestes vetores auxiliares, usam eles em exatamente 5 chamadas (embora nos comentários diga, que são apenas 4??). O resultado volta na posição abaixo do valor de $ra, como usado nos exemplos de aula. Estranhamente, alocam 28 bytes na pilha antes de chamar binary_search, mas usam apenas as 5 posições esperadas. Isto parece um erro na implementação da rotina, ver a análise desta. A interação com o usuário é bem pobre, nenhuma mensagem compreensível, apenas imprime-se cada resultado retornado em cada chamada de rotina. (2,8)	Sem nenhum motivo para isto, iniciam salvando o $ra de retorno a quem chamou a subrotina em $sp+20 (???). Fazem o mesmo processo injustificável com mid (se descobrem que o primeiro teste de fim falha) (???). Estes dois processos violam a ideia básica de comunicação via pilha, pois há entidade usando pilha que não foi criada por ela. E é por isto que a recuperação de valores da pilha está uma meleca completa (embora funcione), com absurdos como buscar valores em $sp+32/$sp+44/$sp+52, oh God! Apesar da confusão no código, a operação de binary_search é recursiva e está funcionando OK. (2,3)	Documentação um tanto pobre, limitou-se a uma descrição geral vaga e as respostas das questões do enunciado. (2,7)	7,80
G5 - BrunoM e Rodrigo	O programa interage com o usuário, mas apenas para pedir a chave, não é possível ao usuário especificar um sub-vetor de pesquisa específico. De resto, o programa principal recebe dados e computa corretamente a posição do elemento escolhido no vetor. O programa tb sai do laço apenasdo jeito certo de acordo com a interação. Como o programa assume que a pesquisa chamada sempre atua no vetor inteiro, ele requer a definição do tamanho do vetor usando um valor n. O código do programa principal está mistura ao código da rotina (parte antes desta e parte depois desta...). (3,5)	Sem nenhum motivo para isto, iniciam salvando o $ra de retorno a quem chamou a subrotina no topo da pilha, o que destrói o valor importante lá colocado por quem a chamou (???). A rotina tem rótulos confusos que atrapalham mais do que ajudam. Compartilhar código de alocação de pilha entre duas chamadas distintas possíveis é uma péssima ideia. Funciona aqui, mas as chances de dar problemas são grandes. (2,3)	Documentação extremamente pobre, não explicaram nada, apenas responderam às questões que valiam 1,0. O código, apesar de escrito de forma confusa está bem comentado. (2,0)	7,80
G6 - Diogo e Giorgia	O programa principal foi desenvolvido de forma rígida demais, deixando os valores Prim, Ult e Chave como constantes carregadas pelo programa principal em registradores no início de sua execução. Para executar o sofware sobre uma chave diferente daquela do código fonte entregue, deve-se editar o código fonte em assembly do MIPS, os valores de "Chave", "Prim" e "Ult", remontar o código executável e re-executar o programa, o que é bem ruim. Ou seja, não há interação com o usuário que executa o programa. Isto torna o programa principal um tanto simples e óbvio, sem que os alunos tenham exercitado o uso mais extenso de chamadas do sistema. Além disto a resposta gerada vem com um texto muito breve, deixando a execução do programa críptica. (2,9)	A subrotina inicia com uma violação direta do processo de comunicação estabelecido na especificação do trabalho, testando valores em registradores ($s1 e $s2) e não valores que ela obtém da pilha. Bem pior que não realizar a comunicação exclusivamente via pilha, a rotina não é recursiva, pois não usa a instrução JAL binary_search dentro do código, apenas usa J, criando um laço com múltiplos pontos de saída do mesmo. Apenas o valor de retorno é comunicado ao programa principal via pilha. (1,0)	Começaram respondendo às questões valendo 1,0 ponto. Claro, como a rotina não é recursiva, as respostas das questões 3/4/5 não fazem sentido. Pelo teor do restante da documentação, fica claro qiue os alunos não capturaram a diferença entre um desvios do fluxo de execução provido por instruções tais como BEQ e J e os desvios provocados por instruções de salto para subrotina, como JAL e BEQZAL. (1,6)	5,50
G7 - Eduardo, Nicolas e Tomás	O programa interage bem com o usuário e recebe corretamente os valores. Mas há dois erros crassos: 1) O uso do registrador $a0 para conter um dos parâmetros. O valor é corretamente gerado, mas ao enviar mensagens para o usuário ele é destruído com ponteiros para textos de uma syscall print string; 2) Na volta da chamada de binary_search retiram o índice da pilha, destroem a pilha e dpois leem de novo o índice da pilha, onde ele não existe mais!!!!Infelizmente, mesmo corrigindo isto, o programa, que tem uma estrutura bem feita, quase nunca retornava resultados corretos, pois há bugs também na subrotina binary_search... (2,7)	De novo, uma boa estrutura de implementação, prejudicada por um erro bobo: calcula-se o endereço correto do elemento a comparar, em $t7, e na hora de usar ele, usam o o endereço errado de $t3... (2,5)	Documentação bem boa do programa, comentários bons. Nas respostas que valiam 1.0, contudo, a resposta da perguntas 3/4/5 não fazem nenhum sentido. A pilha, como é alocada de 20 em 20 bytes, tem que ter um múltiplo de 20 bytes dentro dela durante a execução deste programa. De fato, na terceira chamada recursiva ela deveria estar com 3*20= 60bytes, 28 bytes indica que não sabem como visualizar a situação... (2,8)	8,00
G8 - Enzo, Érico e Luciano	O programa principal foi desenvolvido de forma rígida demais, deixando os valores Prim e Ult gerados de forma fixa e embutidos no código do programa principal, sem sequer definir variáveis na área de dados para estes. A Chave é uma variável definida na área de dados e carregada em registrador no início do programa. Para executar o sofware usando valores de Prim e Ult diferentes daquela do código fonte entregue, deve-se descobrir as linhas em que estes valores são gerados e editar os valores a mão no código fonte em assembly do MIPS. Ainda se deve alterar, se desejado o valor de Chave. Depois ainda se deve remontar o código executável e re-executar o programa, o que é bem ruim. Ou seja, não há nenhuma interação com o usuário que executa o programa. Isto torna o programa principal um tanto simples e óbvio, sem que os alunos tenham exercitado suficientemente o uso de chamadas do sistema. Além disto a resposta gerada vem desacompanhada de qualquer texto explicativo, tornando a execução do programa críptica. (2,5)	A rotina tem código razoavelmente bem implementado, mas possui um erro na parte equivalente ao comando do código C (if (A[mid] == Chave){return mid;}. Quando isto ocorre, sem nenhuma razão (pois ocorre antes de realizar recursão há uma desalocação incorreta de pilha que não foi alocada, gerando erro quando, por exemplo, se faz Chave=15. (2,7)	Documentação por demais sucinta, mas respostas corretas às perrguntas que valiam 1,0 pontos, bons comentários no código. (2,7)	7,90
G9 - GabrielG e LucasC	Implementação bastante confusa e que viola a especificação em muitos aspectos. Ao invés de aceitar um vetor pronto sobre o qual trabalhar, gasta-se muito código obrigando o usuário a entrar o array antes da pesquisa, elemento a elemento. Trabalhor batantes com chamdas dos sistema, mas este tipo de ação é colateral ao problema a resolver. A pesquisa em si nem sempre funciona. A pilha não termina o programa no ponto havia começado, ou seja, o programa deixa lixo na pilha. Aparentemente o vetor é criado na pilha, o que é bizarro e viola a especificação. A interação com o usuário é boa e clara. (2,0)	A subrotina é recursiva, mas não se comunica via pilha apenas, nem com o prorgograma principal, nem com suas chamadas recursivas. Ele não retorna índices de onde um dado elemento se encontra, apenas diz (nem sempre corretamente) se o elemento está ou não na pilha (sic, deveria ser no vetor...). A gerência de pilha é caótica. (1,5)	Documentação bem pobre, as respostas das questões que valiam 1,0 pontos não fazem sentido nos itens 3/4/5. Os mesmos erros conceituais se repetem na documentação e nos comentários do texto. (1,5)	5,00
G10 - GabrielP, LucasB e Marcos	O programa interage bem com o usuário e calcula corretamente o índice do valor procurado, quando este existe e avisa corretamente se o valor nã existe no vetor. A pilha termina o programa exatamente no ponto onde começou, apontando um gerenciamento correto da mesma. Poderiam interagir de forma cíclica com o usuário e não apenas uma vez por execução. (4,0)	A subrotina reflete exatamente o código C como uma recursão implementada da forma canônica. (3,0)	Ótima documentação, sem falhas. (3,0)	10,00
G11 - João e Renato(canc)	O programa principal foi desenvolvido de forma rígida demais, deixando os valores Prim, Ult e Chave como variáveis carregadas pelo programa principal em registradores no início de sua execução. Para executar o sofware sobre uma chave diferente daquela do código fonte entregue, deve-se editar o código fonte em assembly do MIPS, os valores de "Chave", "Prim" e "Ult", remontar o código executável e re-executar o programa, o que é bem ruim. Ou seja, não há interação com o usuário que executa o programa. Isto torna o programa principal um tanto simples e óbvio, sem que os alunos tenham exercitado o uso mais extenso de chamadas do sistema. A única impressão é "Indice: X", onde X é o valor calculado. No mais o programa funciona corretamente e retorna, até onde testado, resultados sempre corretos, deixando a pilha no mesmo estado de antes de sua execução. (3,0)	A subrotina não é recursiva, violando o principal requisito da especificação do trabalho. Existe apenas um laço que implementa o cálculo iterativo usando instruções J ao invés de JAL. Só a chamada a partir do programa principal usa JAL. A subrotina recebe dados via pilha deste, acessa os dados da pilha para fazer a computação, mas não monta qualquer pilha para realizar chamadas recursivas de si própria. (1,5)	Fora uma exposição breve da implementação não-recursiva, apenas respondeu às questões do enunciado, algumas de forma incorreta (algumas não podem ser respondidas se a rotina não é recursiva). Por exemplo, como a subrotina não é recursiva não há uma terceira chamada aninhada da recursão, fazendo com que as respostas das questões 3, 4 e 5 sejam falsas. As telas detalhando a operação da pilha não foram mostradas. (1,5)	6,00
G12 - Leonardo e JenniferMB	Código mal escrito dificulta a compreesão. Melhor seria elaborar processo de geração dos parâmetros da chamada antes, deixando a alocação de pilha e o empilhamento de parâmetros separados, imediatamente antes do "jal binary_search". No retorno da chamada, o código está bem organizado, com o tratamento da pilha e sua destruição imediatamente após a chamada. O programa usa vetores pré-determinados para os parâmetros prim/ult/chave, o que é mais limitado que interagir diretamente com o usuário, mas facilita a escrita do programa. Há 5 elementos nestes vetores auxiliares, usam eles em exatamente 5 chamadas. O resultado volta na posição abaixo do valor de $ra, como usado nos exemplos de aula. A interação com o usuário é bem pobre, nenhuma mensagem compreensível, apenas imprime-se cada resultado retornado em cada chamada de rotina. A operação com os vetores dados é correta. (3,2)	A subrotina é recursiva, apesar do uso de uma estratégia extremamente confusa de organizar o código fonte. No momento em que se descobre que uma recursão será realizada, aloca-se a pilha e carrega-se esta com os dados já disponíveis, seguindo como uma sequência confusa de trechos de código interligados com desvios para definir os parâmetros restantes. Um dos motivos para a confusão é a tentativa injustificável de economizar código, usando apenas um "JAL binary_search" para as duas possíveis chamadas. Note-se, ao usar programação em liguagem de montagem é muitos mais relevante ser claro na programação do que economizar memória. (2,5)	Documentação bastante completa, clara e correta. (3,0)	8,70